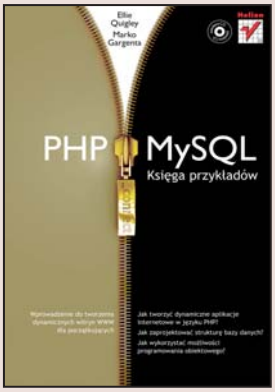
PHP i MySQL. Księga przykładów



Autor: Ellie Quigley, Marko Gargenta
Tłumaczenie: Robert Górczyński
ISBN: 978-83-246-1020-4
Tytuł oryginału: PHP and MySQL by Example
Format: B5, stron: 936
Zawiera CD-ROM

Wprowadzenie do tworzenia dynamicznych witryn WWW dla początkujących

- Jak tworzyć dynamiczne aplikacje internetowe w języku PHP?
- Jak zaprojektować strukturę bazy danych?
- Jak wykorzystać możliwości programowania obiektowego?

Chcesz dodać swojej stronie życia, stworzyć funkcjonalny portal lub sklep internetowy, a może napisać własny blog? Czujesz, że tworzenie atrakcyjnych witryn WWW korzystających z baz danych leży w zasięgu Twoich możliwości? Zatem skorzystaj z potencjału języka PHP i bazy danych MySQL. Technologia ta od dawna cieszy się zasłużoną popularnością wśród webmasterów. Prosta składnia i ogromne możliwości języka PHP w połączeniu ze stabilnością i wydajnością bazy MySQL pozwalają na tworzenie rozbudowanych serwisów i aplikacji internetowych przy relatywnie niskim nakładzie pracy. Ogromnym atutem PHP i MySQL jest fakt, iż są one dostępne nieodpłatnie, na licencji open source.

„PHP i MySQL. Księga przykładów" to doskonałe wprowadzenie do projektowania dynamicznych witryn i aplikacji internetowych z wykorzystaniem tych technologii. Czytając tę książkę, opanujesz podstawy języka PHP, zasady programowania obiektowego, zarządzania sesjami i użytkownikami oraz operacji na plikach. Dowiesz się także, jak projektować i tworzyć bazy danych oraz jak administrować serwerem MySQL. Ponadto nauczysz się zabezpieczać aplikacje WWW przed dostępem osób niepowołanych oraz zastosujesz najlepsze praktyki polecane przez prawdziwych mistrzów programowania.

- Pobieranie i instalowanie PHP i MySQL
- Struktura skryptów PHP
- Podstawowe elementy języka PHP
- Zmienne, stałe i operatory
- Obsługa formularzy
- Operacje na plikach i katalogach
- Wyrażenia regularne
- Praca z serwerem MySQL
- Tworzenie baz i tabel
- Manipulowanie danymi za pomocą poleceń języka SQL
- Połączenie skryptów PHP z bazami danych
- Zarządzanie sesjami i plikami cookie
- Programowanie obiektowe w PHP

Poznaj język PHP i korzystaj z bazy MySQL, tworząc zapierające dech dynamiczne witryny WWW!

Wydawnictwo Helion
ul. Kościuszki 1c
44-100 Gliwice
tel. 032 230 98 63
e-mail: helion@helion.pl

Spis treści

Wstęp **13**

Rozdział 1. Wprowadzenie **15**

1.1. Od statycznych do dynamicznych witryn internetowych 15
1.1.1. Statyczne witryny internetowe 15
1.1.2. Dynamiczne witryny internetowe 16
1.1.3. Czym jest oprogramowanie typu open source? 17
1.2. PHP 18
1.2.1. Skąd pobrać PHP oraz dokumentację? 20
1.3. MySQL 21
1.3.1. Skąd pobrać bazę MySQL oraz dokumentację? 22
1.3.2. Funkcje bazy danych MySQL 23
1.3.3. W jaki sposób zainstalować MySQL i PHP? 24
1.3.4. Zalety wykorzystywania MySQL-a i PHP 24
1.4. Podsumowanie rozdziału 26
1.4.1. Co należy wiedzieć? 26
1.4.2. Co dalej? 26

Rozdział 2. Rozpoczynamy pracę **27**

2.1. Cykl życiowy strony internetowej 27
2.1.1. Analiza strony internetowej 28
2.2. Anatomia skryptu PHP 29
2.2.1. Etapy tworzenia skryptu PHP 29
2.3. Kilka kwestii do przemyślenia 35
2.3.1. PHP i HTML są odmiennymi językami 35
2.3.2. Polecenia, odstępy oraz znaki nowego wiersza 37
2.3.3. Komentarze 39
2.3.4. Używanie funkcji PHP 42
2.4. Przegląd 47
2.4.1. PHP z poziomu wiersza polecenia 47
2.4.2. Wyświetlanie wersji PHP 47
2.4.3. Wykonywanie skryptu z poziomu wiersza polecenia 48
2.4.4. Interaktywne uruchomienie PHP 49
2.4.5. Opcje PHP w wierszu polecenia 50
2.4.6. Plik php.ini 51
2.5. Podsumowanie rozdziału 52
2.5.1. Co należy wiedzieć? 52
2.5.2. Co dalej? 53
Ćwiczenia 53

Rozdział 3. Krótkie wprowadzenie do PHP ... 55
3.1. Szybki start, szybki przewodnik ... 55
3.1.1. Uwaga skierowana do programistów ... 55
3.1.2. Uwaga skierowana do pozostałych ... 55
3.1.3. Składnia PHP i konstrukcje ... 56
3.2. Podsumowanie rozdziału ... 68
3.2.1. Co dalej? ... 68

Rozdział 4. Elementy składowe: rodzaje danych, ciągi tekstowe, zmienne i stałe ... 69
4.1. Rodzaje danych ... 69
4.1.1. Ciągi tekstowe liczb ... 70
4.1.2. Ciągi tekstowe znaków i ujmowanie w znaki cytowania ... 72
4.1.3. Stałe literowe Boolean ... 78
4.1.4. Specjalne rodzaje danych ... 79
4.2. Zmienne ... 81
4.2.1. Definicje i przypisywanie ... 81
4.2.2. Prawidłowe nazwy zmiennych ... 82
4.2.3. Deklarowanie i inicjalizowanie zmiennych ... 82
4.2.4. Wyświetlanie zmiennych ... 87
4.2.5. Zmienne oraz mieszane rodzaje danych ... 90
4.2.6. Łączenie a zmienne ... 91
4.2.7. Odniesienia ... 92
4.2.8. Zmienne zmiennych (zmienne dynamiczne) ... 94
4.2.9. Zasięg zmiennej ... 96
4.2.10. Zarządzanie zmiennymi ... 97
4.2.11. Wprowadzenie do zmiennych formularzy ... 103
4.3. Stałe ... 112
4.3.1. Co to jest stała? ... 112
4.3.2. Tworzenie stałych za pomocą funkcji define() ... 112
4.3.3. Funkcja constant() ... 114
4.3.4. Stałe predefiniowane i „magiczne” ... 115
4.4. Podsumowanie rozdziału ... 116
4.4.1. Co należy wiedzieć? ... 116
4.4.2. Co dalej? ... 117
Ćwiczenia ... 118

Rozdział 5. Operatory ... 121
5.1. Operatory PHP oraz wyrażenia ... 121
5.1.1. Przypisanie ... 122
5.1.2. Pierwszeństwo i reguły łączności ... 122
5.1.3. Operatory arytmetyczne ... 126
5.1.4. Skrócona forma operatorów przypisania ... 128
5.1.5. Operatory automatycznego zwiększenia oraz zmniejszenia o jednostkę ... 130
5.1.6. Niektóre z użytecznych funkcji matematycznych ... 133
5.1.7. Operatory rzutowania ... 134
5.1.8. Operator konkatenacji ... 136
5.1.9. Operatory porównania ... 138
5.1.10. Porównywanie liczb ... 139
5.1.11. Porównywanie ciągów tekstowych ... 141
5.1.12. Operatory logiczne ... 143
5.1.13. Operator warunkowy ... 155
5.1.14. Operatory bitowe ... 158
5.1.15. Operatory uruchamiania ... 162
5.1.16. Operator kontroli błędów ... 162
5.1.17. Operatory rodzajów ... 164

5.2. Podsumowanie rozdziału ... 164
5.2.1. Co należy wiedzieć? ... 164
5.2.2. Co dalej? ... 165
Ćwiczenia ... 165

Rozdział 6. Ciągi tekstowe ... 167
6.1. Co to jest ciąg tekstowy? ... 167
6.1.1. Ujmowanie w cudzysłów ... 168
6.1.2. Operatory ciągów tekstowych ... 172
6.2. Funkcje operujące na ciągach tekstowych ... 175
6.2.1. Formatowanie i wyświetlanie ciągów tekstowych ... 175
6.2.2. Formatowanie liczb oraz wartości pieniężnych ... 181
6.2.3. Określanie długości ciągu tekstowego ... 183
6.2.4. Określanie liczby słów w ciągu tekstowym ... 184
6.2.5. Zmiana wielkości znaków ciągu tekstowego ... 185
6.2.6. Porównywanie ciągów tekstowych ... 188
6.2.7. Wyszukiwanie podobieństw w ciągach tekstowych ... 196
6.2.8. Dzielenie ciągu tekstowego ... 201
6.2.9. Powtarzanie ciągu tekstowego ... 204
6.2.10. Przycinanie i dopełnianie ciągów tekstowych ... 204
6.2.11. Wyszukiwanie i zastępowanie ... 208
6.2.12. Określanie położenia w ciągu tekstowym ... 213
6.2.13. Wyodrębnianie fragmentów ciągu tekstowego — podciągów ... 215
6.2.14. Specjalne ciągi tekstowe oraz znaki ... 224
6.2.15. Praca ze znakami specjalnymi HTML ... 236
6.3. Inne funkcje operujące na ciągach tekstowych ... 240
6.4. Podsumowanie rozdziału ... 244
6.4.1. Co należy wiedzieć? ... 244
6.4.2. Co dalej? ... 244
Ćwiczenia ... 245

Rozdział 7. Instrukcje warunkowe oraz pętle ... 247
7.1. Struktury sterujące, bloki oraz instrukcje złożone ... 247
7.1.1. Instrukcje warunkowe ... 248
7.2. Pętle ... 259
7.2.1. Pętla while ... 259
7.2.2. Pętla do-while ... 261
7.2.3. Pętla for ... 263
7.2.4. Pętla foreach ... 267
7.2.5. Sterowanie działaniem pętli za pomocą poleceń break oraz continue ... 268
7.3. Podsumowanie rozdziału ... 272
7.3.1. Co należy wiedzieć? ... 272
7.3.2. Co dalej? ... 273
Ćwiczenia ... 273

Rozdział 8. Tablice ... 277
8.1. Co to jest tablica? ... 277
8.1.1. Tworzenie tablicy i nadawanie jej nazwy ... 280
8.1.2. Dostęp do elementów tablicy (wartości) ... 286
8.1.3. Wyświetlanie tablicy ... 292
8.1.4. Używanie pętli w celu uzyskania dostępu do elementów tablicy ... 296
8.1.5. Sprawdzenie, czy tablica istnieje ... 304
8.1.6. Tworzenie ciągów tekstowych z tablic oraz tablic z ciągów tekstowych ... 306
8.1.7. Określanie wielkości tablicy ... 309

8.1.8. Wyodrębnianie kluczy oraz wartości z tablic ... 312
8.1.9. Tworzenie zmiennych z elementów tablicy ... 317
8.1.10. Tablice wielowymiarowe ... 325
8.1.11. Sortowanie tablic ... 331
8.1.12. Losowe wybieranie elementów tablicy ... 340
8.2. Modyfikowanie tablic (usuwanie tablicy, usuwanie, dodawanie oraz zmiana elementów) ... 345
8.2.1. Usuwanie tablicy i jej elementów ... 345
8.2.2. Dodawanie elementów do tablicy ... 351
8.2.3. Kopiowanie elementów tablicy ... 356
8.2.4. Łączenie i złączanie tablic ... 358
8.2.5. Operatory tablicy ... 363
8.2.6. Więcej funkcji dotyczących tablic ... 367
8.3. Podsumowanie rozdziału ... 371
8.3.1. Co należy wiedzieć? ... 371
8.3.2. Co dalej? ... 372
Ćwiczenia ... 372

Rozdział 9. Funkcje zdefiniowane przez użytkownika ... 373
9.1. Co to jest funkcja? ... 373
9.1.1. Deklaracja, definicja i wywoływanie funkcji ... 374
9.1.2. Przekazywanie argumentów ... 378
9.1.3. Wartości zwrotne ... 390
9.1.4. Używanie funkcji wywołania zwrotnego ... 398
9.1.5. Zasięg ... 402
9.1.6. Funkcje zagnieżdżone ... 408
9.1.7. Funkcje rekurencyjne ... 411
9.1.8. Biblioteki funkcji — require oraz include ... 414
9.2. Podsumowanie rozdziału ... 418
9.2.1. Co należy wiedzieć? ... 418
9.2.2. Co dalej? ... 418
Ćwiczenia ... 419

Rozdział 10. Więcej o formularzach PHP ... 421
10.1. Wprowadzenie ... 421
10.2. Ogólny opis formularzy HTML ... 421
10.2.1. Rola przeglądarki internetowej ... 422
10.2.2. Rola serwera ... 427
10.2.3. Tworzenie formularzy HTML ... 427
10.3. PHP i formularze ... 433
10.3.1. Dyrektywa register_globals ... 434
10.3.2. Superglobalne tablice PHP służące do pobierania danych formularza ... 435
10.3.3. Request Method ... 436
10.3.4. Silne znaki w nazwach pól formularza oraz danych wejściowych użytkownika ... 436
10.3.5. Parametry formularza i tablica $_REQUEST ... 439
10.3.6. Parametry formularza i styl średni ... 443
10.3.7. Parametry formularza i styl długi (stary) ... 450
10.3.8. Przetwarzanie formularzy z możliwościami wielu wyborów ... 450
10.3.9. Formularze używające przycisków graficznych ... 454
10.3.10. Przetwarzanie formularzy w dokumencie HTML ... 457
10.3.11. Używanie ukrytych pól ... 459
10.3.12. Przekierowanie użytkownika ... 462
10.3.13. Przekazywanie plików ... 466

10.3.14. Trwałe formularze ... 473
10.3.15. Skąd pobrać informacje o tablicach superglobalnych? ... 477
10.3.16. W jaki sposób pobrać informacje o serwerze? ... 478
10.3.17. W jaki sposób pobrać informacje dotyczące środowiska? ... 481
10.4. Podsumowanie rozdziału ... 483
10.4.1. Co należy wiedzieć? ... 483
10.4.2. Co dalej? ... 484

Rozdział 11. Pliki i katalogi ... 485
11.1. Pliki ... 485
11.1.1. Prawa dostępu do plików oraz prawo własności ... 485
11.1.2. Prawa dostępu na platformie Unix/Linux ... 486
11.1.3. Prawa dostępu w systemie Windows ... 489
11.2. Serwer WWW, PHP oraz prawa dostępu ... 490
11.2.1. Wbudowane funkcje PHP ... 491
11.2.2. Uchwyt pliku ... 492
11.2.3. Otworzenie pliku ... 493
11.2.4. Otwieranie pliku w celu odczytu ... 495
11.2.5. Pozycja wewnętrznego wskaźnika pliku ... 504
11.2.6. Otwieranie adresu URL w celu odczytu ... 509
11.2.7. Odczytywanie plików bez korzystania z uchwytu pliku ... 510
11.2.8. Otwieranie pliku w celu zapisu oraz dołączania danych ... 517
11.2.9. Sprawdzanie plików ... 522
11.2.10. Tworzenie, kopiowanie, zmiana nazwy i usuwanie plików ... 527
11.3. Katalogi ... 530
11.3.1. Otwieranie oraz odczyt danych z katalogu ... 530
11.3.2. Pobieranie ścieżki dostępu ... 532
11.3.3. Zmiana i pobieranie bieżącego katalogu roboczego ... 533
11.4. Zarządzanie zawartością za pomocą plików nagłówkowych ... 534
11.4.1. Praktyczny przykład ... 535
11.5. Podsumowanie rozdziału ... 542
11.5.1. Co należy wiedzieć? ... 542
11.5.2. Co dalej? ... 543
Ćwiczenia ... 543

Rozdział 12. Wyrażenia regularne i dopasowanie do wzorca ... 545
12.1. Co to jest wyrażenie regularne? ... 545
12.2. Funkcje dopasowania do wzorca ... 547
12.2.1. Wyszukiwanie wzorca ... 548
12.2.2. Wyszukiwanie i zastępowanie ... 555
12.2.3. Przejmowanie kontroli — metaznaki regex ... 567
12.2.4. Wyszukiwanie wzorca w pliku tekstowym ... 601
12.2.5. Sprawdzanie poprawności formularzy za pomocą PHP ... 604
12.2.6. Pomoc w internecie ... 611
12.3. Podsumowanie rozdziału ... 612
12.3.1. Co należy wiedzieć? ... 613
12.3.2. Co dalej? ... 613
Ćwiczenia ... 613

Rozdział 13. Wprowadzenie do MySQL-a ... 615
13.1. Ogólny opis baz danych ... 615
13.1.1. Bazy danych typu klient-serwer ... 616
13.1.2. Przesyłanie komunikatów do bazy danych ... 617
13.1.3. Mocne i słabe strony MySQL-a ... 617

13.2. Anatomia relacyjnej bazy danych ... 619
13.2.1. Serwer bazy danych ... 619
13.2.2. Baza danych ... 620
13.2.3. Tabele ... 620
13.2.4. Rekordy oraz pola ... 621
13.2.5. Klucz podstawowy i indeksy ... 622
13.2.6. Schemat bazy danych ... 623
13.3. Nawiązywanie połączenia z bazą danych ... 623
13.3.1. Opcje klienta wiersza polecenia mysql ... 625
13.3.2. Narzędzia z graficznym interfejsem użytkownika ... 627
13.4. System praw dostępu bazy danych MySQL ... 630
13.4.1. Logowanie do serwera bazy danych ... 631
13.4.2. Określanie dostępnych baz danych ... 632
13.4.3. Tabela user ... 634
13.4.4. Tabela db ... 635
13.4.5. Tabela host ... 636
13.4.6. Rzeczywisty przykład ... 637
13.4.7. Nadawanie i odbieranie praw ... 638
13.4.8. Tworzenie i usuwanie bazy danych ... 639
13.4.9. Niektóre użyteczne funkcje MySQL-a ... 641
13.5. Podsumowanie rozdziału ... 642
13.5.1. Co należy wiedzieć? ... 642
13.5.2. Co dalej? ... 643

Rozdział 14. Wprowadzenie do języka SQL ... 645
14.1. Co to jest SQL? ... 645
14.1.1. Standaryzacja SQL ... 646
14.1.2. Wykonywanie poleceń SQL ... 646
14.1.3. Informacje o poleceniach i zapytaniach SQL-a ... 648
14.1.4. SQL i baza danych ... 651
14.1.5. Tabele bazy danych SQL ... 652
14.2. SQL Data Manipulation Language (DML) ... 653
14.2.1. Polecenie SELECT ... 654
14.2.2. Polecenie INSERT ... 667
14.2.3. Polecenie UPDATE ... 668
14.2.4. Polecenie DELETE ... 669
14.3. SQL Data Definition Language ... 670
14.3.1. Utworzenie bazy danych ... 670
14.3.2. Rodzaje danych SQL ... 671
14.3.3. Tworzenie tabeli ... 671
14.3.4. Tworzenie klucza ... 675
14.3.5. Związki ... 677
14.3.6. Modyfikacja tabeli ... 681
14.3.7. Usuwanie tabeli ... 682
14.3.8. Usuwanie bazy danych ... 682
14.4. Funkcje SQL ... 682
14.4.1. Funkcje liczbowe ... 683
14.4.2. Funkcje operujące na ciągach tekstowych ... 686
14.4.3. Funkcje daty i godziny ... 687
14.5. Podsumowanie rozdziału ... 690
14.5.1. Co należy wiedzieć? ... 690
14.5.2. Co dalej? ... 691
Ćwiczenia ... 691

Rozdział 15. Integracja PHP oraz MySQL-a 695
15.1. Wprowadzenie 695
15.1.1. Nawiązanie połączenia z serwerem bazy danych 695
15.1.2. Wybór bazy danych 698
15.1.3. Wykonywanie poleceń SQL (INSERT, UPDATE, DELETE) 699
15.1.4. Pobieranie wyników zapytania (SELECT) 702
15.1.5. Inne użyteczne funkcje MySQL-a 706
15.2. Przykład w postaci księgi gości 711
15.2.1. Krok 1. — projekt bazy danych 712
15.2.2. Krok 2. — strona pozwalająca umieścić komentarz 713
15.2.3. Krok 3. — wyświetlenie wszystkich komentarzy 717
15.2.4. Podsumowanie przykładu 718
15.3. Podsumowanie rozdziału 719
15.3.1. Co należy wiedzieć? 719
15.3.2. Co dalej? 719
Ćwiczenia 719

Rozdział 16. Mechanizm cookies oraz sesje 721
16.1. Co oznacza termin bezstanowy? 721
16.2. Czym są pliki cookie? 722
16.2.1. Składniki pliku cookie 723
16.2.2. Atrybuty cookie 724
16.3. PHP i mechanizm cookies 725
16.3.1. Tworzenie cookie za pomocą funkcji setcookie() 725
16.3.2. Śledzenie użytkowników witryny za pomocą mechanizmu cookies 732
16.3.3. Wydłużenie okresu ważności cookie 737
16.3.4. Buforowanie a nagłówki HTTP 738
16.3.5. Usuwanie pliku cookie 742
16.4. Co to jest sesja? 742
16.4.1. Gdzie przechowywać sesje? 745
16.4.2. Uruchomienie sesji bazującej na plikach cookie 746
16.4.3. Rejestrowanie sesji 748
16.4.4. Zapisywanie tablic w sesji 752
16.4.5. Funkcje cookie sesji oraz opcje konfiguracyjne 756
16.4.6. Dokonywanie ustawień za pomocą sesji 757
16.4.7. Nazywanie sesji 765
16.4.8. Sesje bez plików cookie 766
16.4.9. Przekazywanie identyfikatora sesji za pomocą łącza 771
16.4.10. Zmiana identyfikatora sesji 776
16.4.11. Zakończenie sesji 778
16.4.12. Konfiguracja sesji w trakcie działania programu 780
16.4.13. Implementacja systemu logowania z użyciem sesji 782
16.5. Podsumowanie rozdziału 787
16.5.1. Co należy wiedzieć? 788
16.5.2. Co dalej? 788
Ćwiczenia 789

Rozdział 17. Obiekty 791
17.1. Co to są obiekty? 791
17.1.1. Obiekty i klasy 792
17.2. Praca z klasami 793
17.2.1. Definicja klasy 793
17.2.2. Ustanawianie egzemplarza klasy 794
17.2.3. Tworzenie kompletnej klasy 797

17.2.4. Wyświetlanie obiektu ... 800
17.2.5. Funkcje pobierające informacje o klasie ... 801
17.2.6. Hermetyzacja oraz ukrywanie informacji ... 801
17.2.7. Elementy klasy oraz zasięg ... 802
17.2.8. Metody magiczne ... 805
17.2.9. Dziedziczenie ... 815
17.2.10. Nadpisywanie metod ... 819
17.2.11. Dostęp chroniony ... 821
17.3. Niektóre funkcje obiektowe w PHP 5 ... 825
17.3.1. Klasy i metody final ... 825
17.3.2. Elementy statyczne ... 826
17.3.3. Stałe w klasie ... 828
17.3.4. Ponowne użycie klasy ... 829
17.4. Podsumowanie rozdziału ... 834
17.4.1. Co należy wiedzieć? ... 834
Ćwiczenia ... 835

Dodatek A Tworzenie galerii sztuki ... 837
A.1. Ogólny opis projektu ... 837
A.2. Strony publiczne i prywatne ... 837
A.3. Tworzenie witryny internetowej ... 838
A.3.1. Tworzenie bazy danych ... 839
A.3.2. Strony administracyjne ... 840
A.3.3. Strony publiczne ... 850
A.3.4. Ochrona stron za pomocą logowania ... 856
A.4. Instalacja aplikacji galerii sztuki ... 858
A.4.1. Gdzie znajdują się pliki witryny Canvas Gallery? ... 858
A.4.2. Instalacja bazy danych MySQL ... 859
A.4.3. Edycja stron PHP ... 861
A.5. Podsumowanie ... 861

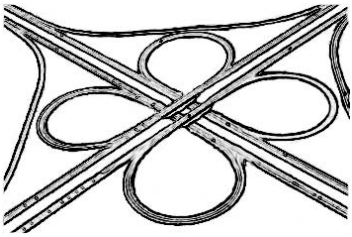
Dodatek B PHP i wiadomości e-mail ... 863
B.1. Serwer poczty ... 863
B.2. MIME (Multipurpose Internet Mail Extensions) ... 864
B.3. Opcje środowiska uruchomieniowego ... 864
B.4. Funkcja mail() ... 865
B.5. Wysyłanie prostej wiadomości e-mail ... 866
B.6. Przykład — wysyłanie wiadomości HTML ... 867
B.6.1. Wysyłanie wiadomości e-mail z załącznikiem ... 869

Dodatek C PHP oraz funkcje daty i godziny ... 873
C.1. Formatowanie daty i godziny ... 873
C.1.1. Funkcja date() ... 873
C.1.2. Funkcja strftime() ... 876
C.2. Pobieranie znacznika czasu ... 878
C.2.1. Funkcja time() ... 878
C.2.2. Funkcja mktime() ... 879
C.2.3. Tworzenie znacznika czasu systemu Unix z ciągu tekstowego ... 881
C.3. Pobieranie daty i godziny ... 882
C.3.1. Sprawdzanie poprawności daty ... 884

Dodatek D Bezpieczeństwo oraz usuwanie błędów ... 885
D.1. Bezpieczeństwo ... 885
D.1.1. Bezpieczeństwo sieci oraz protokół SSL ... 885
D.1.2. Bezpieczeństwo systemu operacyjnego, serwera WWW oraz systemu plików ... 886
D.2. Zabezpieczanie PHP oraz MySQL ... 887
D.2.1. Podstawowe zasady bezpieczeństwa w PHP ... 887
D.3. Usuwanie błędów ... 893
D.3.1. Włączenie komunikatów błędów ... 893
D.3.2. Usuwanie błędów składni w pierwszej kolejności ... 894
D.3.3. Polecenia diagnostyczne print ... 895
D.3.4. Usuwanie błędów SQL ... 898
D.3.5. Więcej informacji dotyczących usuwania błędów ... 899

Dodatek E Procedury instalacyjne ... 901
E.1. Informacje o serwerach WWW ... 901
E.2. Instalacja serwera Apache w systemie Windows ... 902
E.3. Instalacja PHP w systemie Windows ... 903
E.4. Instalacja PHP w systemach Linux/Unix ... 903
E.5. Instalacja PHP w systemie Mac OS X ... 904
E.6. Konfiguracja serwera Apache do pracy z modułem PHP (wszystkie platformy) ... 904
E.6.1. Sprawdzanie instalacji PHP oraz Apache ... 904
E.7. Konfiguracja pliku php.ini (wszystkie platformy) ... 905
E.8. Instalacja MySQL-a w systemie Windows ... 907
E.9. Instalacja MySQL-a w systemach Linux/Unix ... 907
E.10. Instalacja MySQL-a w systemie Mac OS X ... 907
E.11. Przeczytaj podręcznik ... 907

Skorowidz ... 909



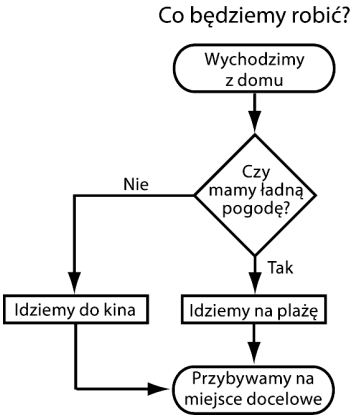
Rozdział 7.

Instrukcje warunkowe oraz pętle

7.1. Struktury sterujące, bloki oraz instrukcje złożone

Na rysunku 7.1 został pokazany schemat przykładowego programu. Pojęcie schematu oznacza graficzne przedstawienie elementów projektu. Jest to pomocne podczas uświadamiania sobie, jakie decyzje muszą zostać podjęte w celu realizacji danego zadania. Ludzie kontrolują swoje życie, podejmując decyzje, podobnie dzieje się w programach. W rzeczywistości, według podręczników technologii komputerowych, dobry język programowania pozwala na sterowanie działaniem programu na trzy sposoby:

- Wykonywanie sekwencji poleceń
- Odgałęzienie do alternatywnej sekwencji poleceń na podstawie dokonanego wyboru
- Powtarzanie określonej sekwencji poleceń, dopóki nie zostanie spełniony warunek



Rysunek 7.1. *Schemat przykładowego programu*

PHP musi być więc dobrym językiem programowania.

Używaliśmy już programów, które wykonywały sekwencje poleceń, jedno za drugim. W tej chwili przeanalizujemy rozgałęzianie oraz struktury kontrolne pętli, które pozwalają na sterowanie działaniem i dokonywanie zmian w zależności od spełnienia pewnych wyrażeń warunkowych.

Konstrukcje podejmowania decyzji (`if`, `if-else`, `if -else if`) zawierają wyrażenia kontrolne określające wykonywany blok poleceń. Konstrukcje w postaci pętli (`while`, `for`) pozwalają programowi na powtarzanie wykonywania bloku poleceń aż do chwili spełnienia danego warunku.

Instrukcje złożone, inaczej nazywane *blokiem*, składają się z poleceń lub grupy poleceń umieszczonych w nawiasach klamrowych. Składniowo blok jest odpowiednikiem pojedynczego polecenia i zwykle znajduje się za konstrukcjami `if`, `else`, `while` lub `for`. Przykład bloku:

```
{polecenie; polecenie; polecenie}
```

7.1.1. Instrukcje warunkowe

Konstrukcje warunkowe sterują działaniem programu. Jeżeli warunek jest prawdziwy, wtedy program wykona blok poleceń. Natomiast jeśli warunek nie jest spełniony, wówczas program wykona inny blok poleceń. Konstrukcje podejmowania decyzji (`if`, `else`, `switch`) zawierają wyrażenia kontrolne, które określają, czy dany blok zostanie wykonany. Spełnienie warunku po poleceniu `if` powoduje zwrócenie wartości `true` oraz wykonanie bloku znajdującego się za poleceniem. W przeciwnym razie wartością zwrotną jest `false`, a blok nie zostaje wykonany.

FORMAT KONSTRUKCJI `IF`:

```
if (warunek) {
   polecenia;
}
```

Przykład:

```
if ($wiek > 21) {
   print 'Chodźmy na przyjęcie!';
}
```

Blok poleceń (lub pojedyncze polecenie) jest umieszczony w nawiasach klamrowych. Polecenia są zazwyczaj wykonywane po kolei. Jeżeli po instrukcji warunkowej znajduje się tylko jedno polecenie, wówczas nawiasy klamrowe są opcjonalne.

Konstrukcja `if-else`. „Będzie lepiej, jeśli zapłacisz mi teraz lub…”. Czytelnik zapewne słyszał już takie zdania. Polecenia PHP mogą być obsługiwane w taki sam sposób za pomocą rozgałęziającej struktury `if-else`. Taka konstrukcja pozwala na podjęcie jednej z dwóch decyzji. Instrukcja `if` sprawdza pierwsze wyrażenie warunkowe w nawiasach i jeśli wyrażenie zwraca wartość `true`, wówczas zostanie wykonany blok

poleceń znajdujący się w nawiasach klamrowych za instrukcją `if`. W przeciwnym przypadku zostanie wykonany blok poleceń znajdujący się po instrukcji `else`. Blok `else` jest opcjonalny. Zobacz listing 7.1.

FORMAT KONSTRUKCJI IF-ELSE:

```
if (warunek) {
   polecenia1;
}
else {
   polecenia2;
}
```

Przykład:

```
if ($x > $y) {
   print "Zmienna $x jest większa.";
}
else
{
   print "Zmienna $y jest większa.";
}
```

LISTING 7.1.

```
(Formularz HTML)
    <html><head><title>Opłata za przejazd</title></head>
    <body bgcolor="lightgreen">
    <font face="arial" size="+1">
    <form method="get" action="listing7.1a.php">
    <p>Podaj swój wiek:
    <input type="text" name="age" size=2>
    <p>
    <input type="submit" name="submit_age" value="Kup bilet" >
    </form>
    </body>
    </html>
--------------------------------------------------------------------------------
(Skrypt PHP)
    <html><head><title>Opłata za przejazd</title></head>
    <body bgcolor="lightgreen">
    <font face="arial" size="+1">
    <p>
    <?php

       extract($_REQUEST); // Używanie funkcji extract wymaga zachowania ostrożności
(1)       if ( ! isset ( $submit_age )){ // Prosta instrukcja warunkowa.
             exit;
          }

    ?>
    <table border="1" cellpadding="10"><tr bgcolor="yellow">
    <?php

(2)     if ( $age >= 55 ){
(3)        $price = 8.25;
```

```
(4)        print "<td><b>Opłata za przejazd emeryta wynosi $price zł!</td>";
(5)     }
(6)     else {
(7)        $price = 10.00;
(8)        print "<td><b>Opłata za przejazd osoby dorosłej wynosi $price zł.</td>";
        }

    ?>
    </body></html>
```

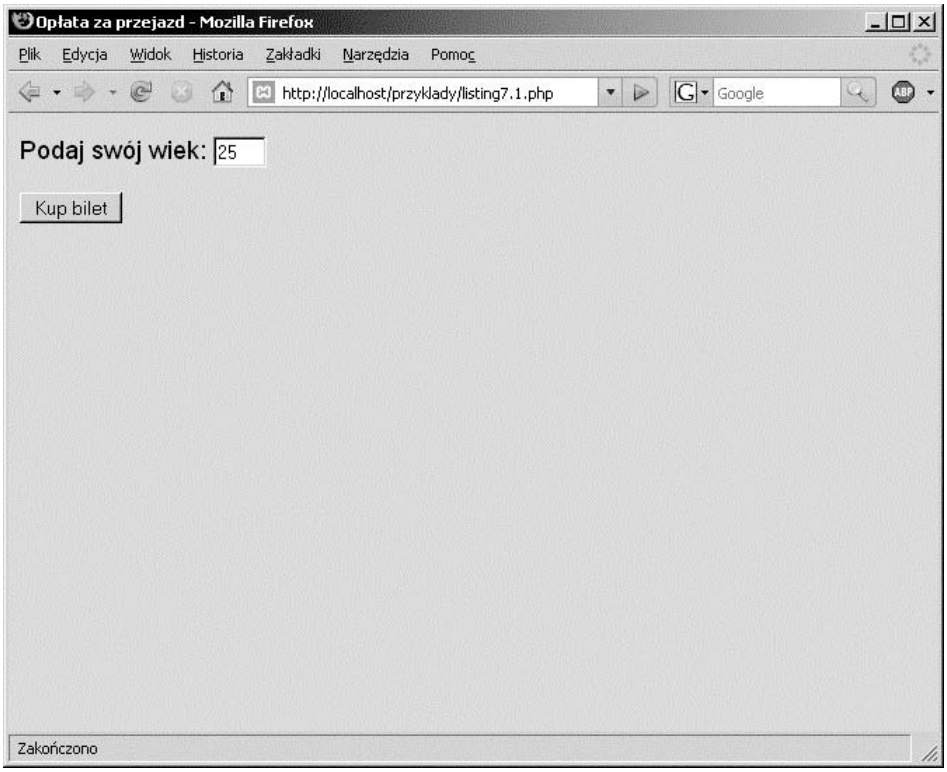
OBJAŚNIENIE

(1) Jeżeli zmienna nie zostanie ustalona, to program zakończy działanie.

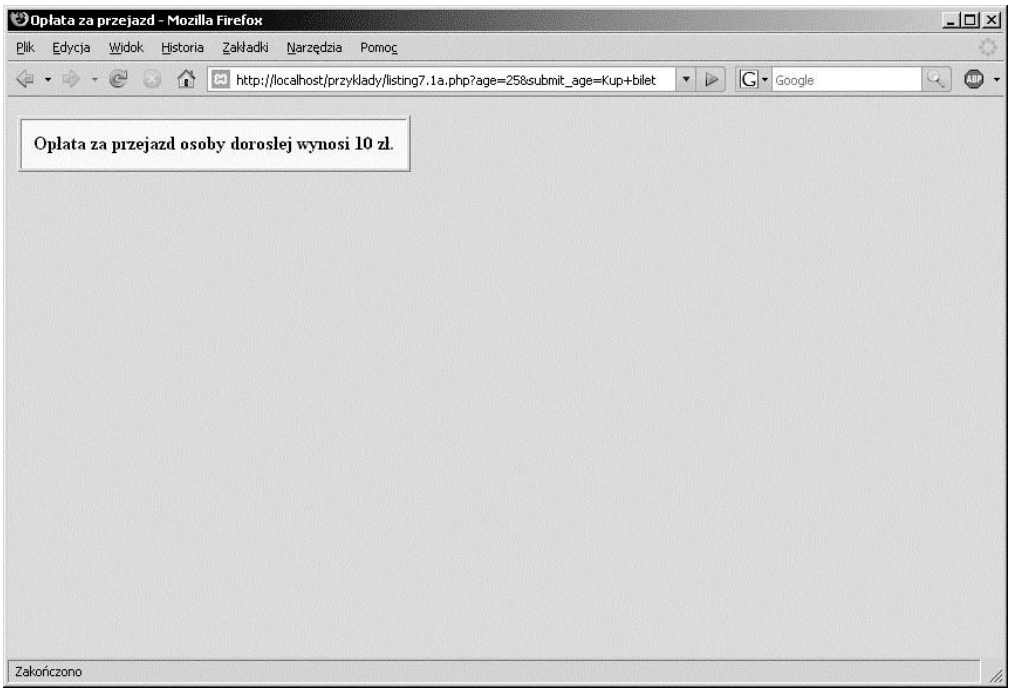
(2 – 4) Jeżeli wartość zmiennej `$age` jest większa lub równa `55`, nastąpi wykonanie wierszy trzeciego i czwartego. Zobacz rysunki 7.2 oraz 7.3.

(5) Ten zamykający nawias klamrowy kończy blok poleceń po instrukcji `if`. Ponieważ w bloku znajduje się tylko jedno polecenie, nawiasy klamrowe nie są wymagane.

(6 – 8) Polecenia bloku `else` (wiersze 7. oraz 8.) są wykonywane, gdy instrukcja `if` w wierszu drugim zwróci wartość `false`.



Rysunek 7.2. *Formularz HTML przedstawiony na listingu 7.1*



Rysunek 7.3. *Po naciśnięciu przycisku „Kup bilet" nastąpi wysłanie formularza. Rysunek przedstawia dane wyjściowe skryptu PHP listingu 7.1*

Konstrukcja `if-elseif`. „Jeśli dostaniesz złotówkę, możesz pójść do sklepu, w którym wszystko jest za złotówkę, lub jeśli dostaniesz 10 zł, wtedy możesz kupić kilka filmów, lub jeśli dostaniesz 20 zł, wtedy możesz kupić płytę CD… lub zapomnij o tym!". PHP dostarcza również inną konstrukcję służącą do podejmowania decyzji `if-elseif`. Pozwala ona na utworzenie struktury decyzji z wieloma możliwościami wyboru.

FORMAT KONSTRUKCJI `IF-ELSEIF`:

```
if (warunek) {
   polecenia1;
}
elseif (warunek)
   polecenia2;
}
elseif (warunek) {
   polecenia3;
}
else {
   polecenia4;
}
```

Jeżeli pierwsze wyrażenie warunkowe znajdujące się za słowem kluczowym `if` przyjmie wartość `true`, wówczas nastąpi wykonanie znajdującego się za nim bloku poleceń, a działanie programu będzie kontynuowane od poleceń znajdujących się po bloku `else`. W przeciwnym przypadku, jeśli wyrażenie warunkowe po słowie kluczowym

przyjmie wartość `false`, nastąpi przejście do pierwszej instrukcji `elseif` i obliczenie jej wartości. Wartość `true` tej instrukcji spowoduje wykonanie znajdującego się za nią bloku instrukcji, natomiast wartość `false` powoduje sprawdzenie kolejnej instrukcji `elseif`. Jeżeli po sprawdzeniu wszystkich instrukcji `elseif` żadna nie zwróci wartości `true`, wtedy kontrola nad programem przechodzi do instrukcji `else`. Mimo że blok `else` nie jest wymagany, to funkcjonuje jako działanie domyślne w przypadku, gdy wszystkie poprzednie instrukcje warunkowe zwróciły wartość `false`. Przykład zastosowania konstrukcji `if-elseif` został przedstawiony na listingu 7.2. Dane wyjściowe listingu pokazano na rysunkach 7.4 oraz 7.5.

LISTING 7.2.

```
(Formularz HTML)
    <html>
    <head><title>Opłata za przejazd</title></head>
    <body bgcolor="azure">
(1) <form method="get" action="listing7.2a.php">
       <p>Podaj swój wiek:<br />
(2)    <input type="text" name="age" size=3 />
       <p>
(3)    <input type="submit" name="submit_fare" value="Kup bilet" />
    </form>
    </body>
    </html>
-------------------------------------------------------------------------------
(Skrypt PHP)
    <html><head><title>Opłata za przejazd</title></head>
    <body bgcolor="chartreuse">
    <font face="arial" size="+1">
(4) <?php
       extract($_REQUEST); // Pobranie danych wejściowych formularza.
         // Używanie tablicy REQUEST wymaga zachowania ostrożności
(5)    if ( ! isset ($submit_fare) || $age == '' ){
          print 'Należy podać swój wiek..<br />';
       exit;
       }
    ?>
    <table border="1" cellpadding="10">
    <tr bgcolor="azure">
    <?php

(6)    if ($age > 0 && $age < 13) {
          $price = 5.00;
          print "<td><b>Opłata za przejazd dziecka wynosi $price zł!</td>";
       }
(7)    elseif ($age >= 13 && $age < 55 ){
          $price = 8.25;
          print "<td><b>Opłata za przejzad osoby dorosłej wynosi $price zł.</td>";
       }
(8)    elseif ( $age >= 55 && $age <= 120){
          $price = 10.00;
          print "<td><b>Opłata za przejazd emeryta wynosi $price zł.</td>";
       }
```

```
(9)     else {
           print '<td><b>Nie jesteś człowiekiem!</td>';
        }
(10)  ?>
     </tr>
     </table>
     </font>
     </body>
     </html>
```

OBJAŚNIENIE

(1) W tym wierszu rozpoczyna się formularz HTML. Atrybutowi `action` została przypisana nazwa skryptu, który będzie wykonany po wysłaniu formularza.

(2) Pole tekstowe o nazwie `age` będzie przechowywało trzy znaki.

(3) Dane wejściowe formularza HTML mają nadaną nazwę `submit_fare`, która będzie używana w skrypcie PHP.

(4) Otwierający znacznik PHP nakazuje PHP rozpoczęcie przetwarzania.

(5) Przyglądamy się poleceniom PHP. Jeżeli użytkownik naciśnie przycisk wysyłający formularz, wtedy nastąpi wykonanie pliku skryptu. Nazwy urządzeń wejściowych są przechowywane przez PHP w zmiennych `$age` oraz `$submit_fare`. Wyrażenie warunkowe sprawdza, czy wymienione zmienne zostały ustalone. Jeśli nie są ustawione, oznacza to, że albo formularz nie został wysłany, albo użytkownik pozostawił niewypełnione pola, albo jedno i drugie.

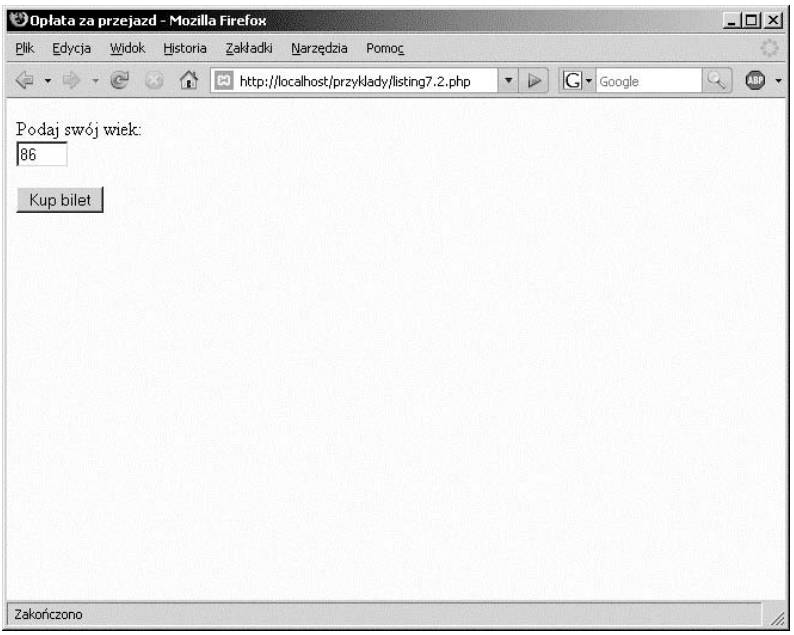
(6) Jeśli wiek użytkownika jest większy niż `0` i mniejszy od `13`, wtedy nastąpi wykonanie bloku poleceń po sprawdzającej ten warunek instrukcji `if`. Po wykonaniu bloku program będzie kontynuował swoje działanie od dziesiątego wiersza. Jeśli sprawdzany warunek zwróci wartość `false`, nastąpi przejście do wiersza siódmego.

(7) Jeżeli instrukcja warunkowa w wierszu szóstym zwróci wartość `false`, wtedy program sprawdzi, czy prawdziwe jest wyrażenie warunkowe znajdujące się po instrukcji `elseif`. Jeśli wspomniane wyrażenie zwróci wartość `true` (na przykład podany wiek będzie większy lub równy `13`, ale mniejszy od `55`), wówczas nastąpi wykonanie bloku poleceń. W przeciwnym przypadku program przejdzie do wiersza ósmego.

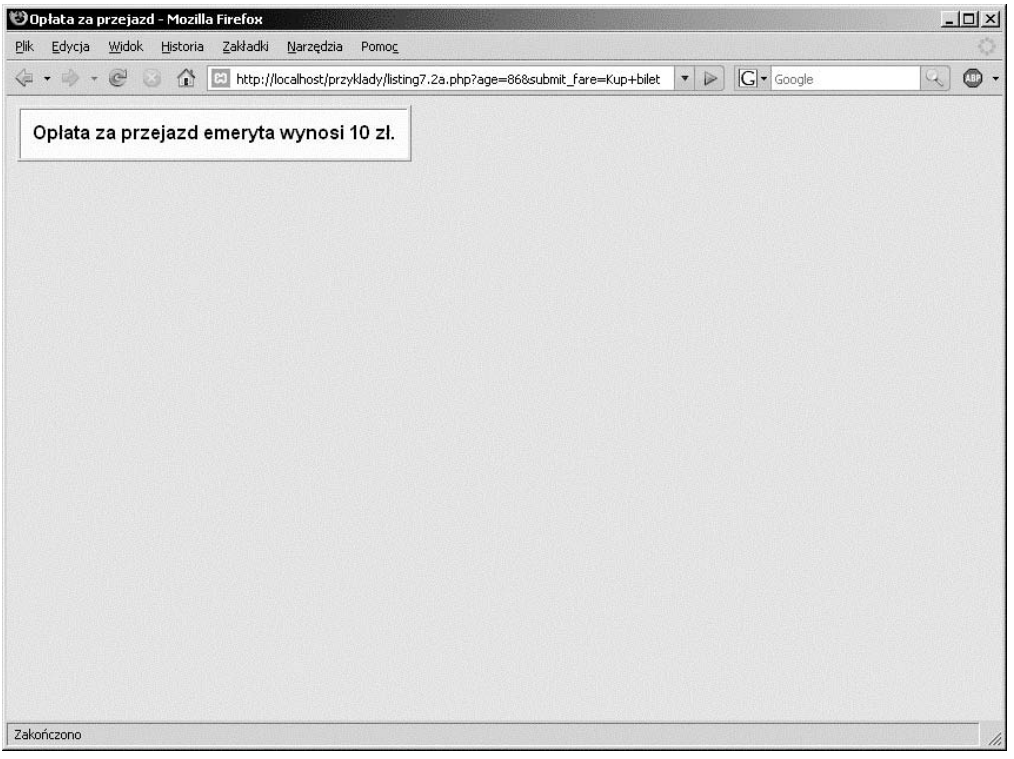
(8) Jeżeli instrukcja warunkowa w wierszu siódmym zwróci wartość `false`, nastąpi sprawdzenie tej instrukcji `elseif`. Wartość `true` spowoduje wykonanie bloku poleceń znajdującego się za instrukcją. W przeciwnym przypadku, nastąpi wykonanie bloku poleceń znajdującego się za instrukcją `else` w wierszu dziewiątym.

(9) Jeśli żadna z powyższych instrukcji warunkowych nie zwróciła wartości `true`, wtedy kontrola nad programem przechodzi do bloku `else`, często nazywanego warunkiem domyślnym, i następuje wykonanie poleceń z tego bloku.

(10) Zamykający znacznik PHP, interpreter kończy tutaj swoją pracę.



Rysunek 7.4. *Formularz HTML przedstawiony na listingu 7.2*



Rysunek 7.5. *Dane wyjściowe skryptu PHP z listingu 7.2*

Konstrukcja `switch`. Konstrukcja `switch` jest alternatywą konstrukcji warunkowej `if-elseif`, często nazywaną *konstrukcją wariantową*, i powoduje, że przy większej liczbie opcji program jest łatwiejszy w odczycie.

FORMAT KONSTRUKCJI SWITCH:

```
switch (wyrażenie) {
   case etykieta:
      polecenie(a);
      break;
   case etykieta:
      polecenie(a);
      break;
      ...
   default : polecenie;
}
```

Przykład:

```
switch ($kolor) {
   case 'czerwony':
      print 'Gorący!';
      break;
   case 'niebieski':
      print 'Zimny.';
      break;
   default:
      print 'To nie jest dobry wybór.';
      break;
}
```

Wartość wyrażenia `switch` jest dopasowywana względem wyrażeń nazwanych etykieta po słowie kluczowym `case`. Etykiety słowa `case` są stałymi w postaci albo ciągu tekstowego, albo liczb. Każda etykieta kończy się średnikiem. Etykieta domyślna (`default`) jest opcjonalna, ale jej wykonanie następuje wtedy, gdy żadna z wcześniejszych etykiet nie została dopasowana do wyrażenia `switch`. Po znalezieniu dopasowania następuje wykonanie poleceń znajdujących się po dopasowanej etykiecie. Jeśli żadna etykieta nie zostanie dopasowana, wówczas kontrola nad programem przechodzi do etykiety `default`, która jest opcjonalna. Pominięcie polecenia `break` spowoduje, że wykonane zostaną wszystkie polecenia znajdujące się za dopasowaną etykietą aż do napotkania polecenia `break` lub opuszczenia całego bloku `switch`. Zapoznaj się z listingiem 7.3.

LISTING 7.3.

```
(Formularz HTML)
    <html>
    <head>
    <title>Wybierz kolor czcionki</title>
    </head>
    <body bgcolor="9BCD93">
    <font face="arial" >
```

```
    <b>
    <form method="get" action="listing7.3a.php">
       <br />Wybierz kolor czcionki:
       <br /><input type="radio" name="color" value="red" /> czerwony
       <br /><input type="radio" name="color" value="blue"/> niebieski
       <br /><input type="radio" name="color" value="purple" /> fioletowy
       <br /><input type="radio" name="color" value="green" /> zielony
       <p>
       <input type="submit" name="submit_color" value="Wybierz kolor" />
    </form>
    </b>
    </body>
    </html>
-------------------------------------------------------------------------------
(Skrypt PHP)
    <html><head><title>Kolor czcionki</title></head>
    <body bgcolor="lightgreen">
    <font face="arial" size="+1">
    <p>
(1) <?php
       extract($_REQUEST);
         // Używanie tablicy REQUEST wymaga zachowania ostrożności
(2)    if (! isset($submit_color)){ // Sprawdzenie, czy zmienne zostały ustalone.
          exit;
       }
    ?>
    <table border="2" cellpadding="10">
    <tr bgcolor="white">
(3) <?php
(4)    switch ( $color ) {
(5)    case 'red':
          print "<td><b><font color=".$color.">Czcionka jest czerwona</td>";
(6)       break;
(7)    case 'blue':
          print "<td><b><font color=".$color.">Czcionka jest niebieska</td>";
          break;
       case 'purple':
          print "<td><b><font color=".$color.">Czcionka jest fioletowa</td>";
          break;
       case 'green':
          print "<td><b><font color=".$color.">Czcionka jest zielona</td>";
          break;
(8)    default:
          print "<td><b><font color='.'black'.'>Czcionka jest czarna</td>";
          break;
(9)    }

    ?>
    </tr>
    </table>
    </font>
    </body>
    </html>
```

OBJAŚNIENIE

(1) W tym wierszu rozpoczyna się skrypt PHP.

(2) Jeżeli zmienna `$submit_color` nie jest ustalona, oznacza to, że formularz nie został wysłany.

(3) W tym wierszu rozpoczyna się program PHP.

(4) W wyrażeniu `switch` wartość zmiennej `$color` jest dopasowywana względem wartości każdej z poniższych etykiet `case`. (PHP nadaje zmiennej `$color` taką samą nazwę jak nazwa nadana przyciskowi opcji oraz przypisuje zmiennej wartość, która została wybrana przez użytkownika po kliknięciu przycisku. Jeżeli użytkownik kliknie `czerwony`, wtedy wartość zmiennej `$color` wynosi `red`). Zobacz rysunki 7.6 oraz 7.7.

(5) Pierwszą sprawdzaną etykietą `case` jest `red`. Jeżeli użytkownik kliknął przycisk opcji „czerwony", wówczas kolorem czcionki będzie czerwony i nastąpi wyświetlenie w komórce tabeli komunikatu „Czcionka jest czerwona".

(6) Polecenie `break` powoduje kontynuowanie wykonywania programu po wierszu dziewiątym. Bez tego polecenia, program kontynuowałby wykonywanie kolejnych poleceń aż do napotkania polecenia `break` lub opuszczenia konstrukcji `switch`. Takie zachowanie programu nie jest przez nas pożądane.

(7) Pierwszą sprawdzaną etykietą `case` jest `red`. Jeżeli użytkownik kliknął przycisk opcji „niebieski", wtedy PHP przeskoczy etykietę `red` i przejdzie do kolejnej, którą jest `blue`. Ponieważ wartość tej etykiety zostanie dopasowana do zmiennej `$color`, to kolorem czcionki będzie niebieski, a w komórce tabeli nastąpi wyświetlenie komunikatu „Czcionka jest niebieska".

Polecenie `break` powoduje przekazanie kontroli nad programem do wiersza dziewiątego.

Jeżeli etykiety `red` oraz `blue` nie zostaną dopasowane do koloru zmiennej `$color`, wtedy nastąpi sprawdzenie etykiety `purple`. Dopasowanie spowoduje wykonanie poleceń z dopasowanego bloku.

Polecenie `break` powoduje przekazanie kontroli nad programem do wiersza dziewiątego.

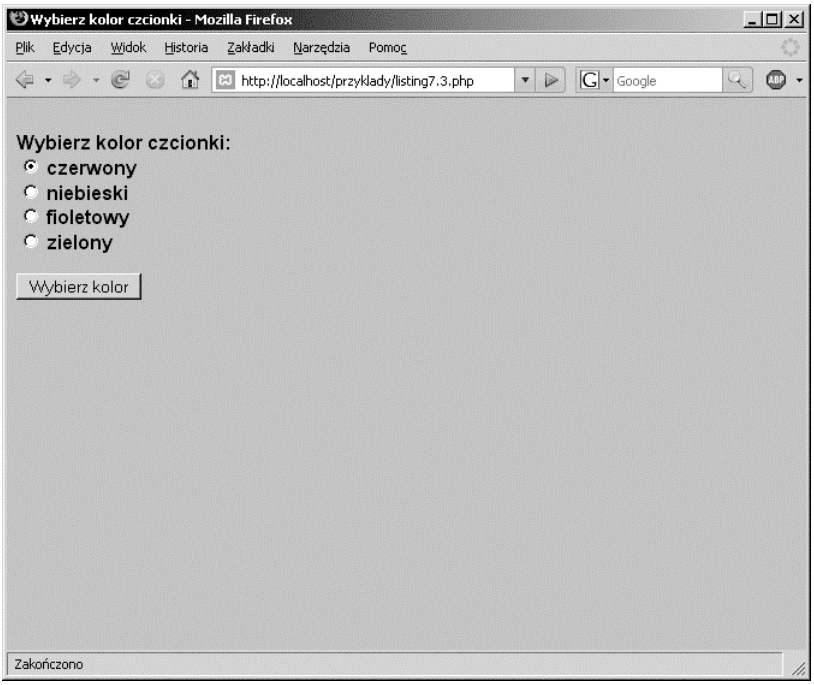
Jeżeli etykiety `red`, `blue` oraz `purple` nie zostaną dopasowane do koloru zmiennej `$color`, wtedy nastąpi sprawdzenie etykiety `green`. Dopasowanie spowoduje wykonanie poleceń z dopasowanego bloku.

Polecenie `break` powoduje przekazanie kontroli nad programem do wiersza dziewiątego.

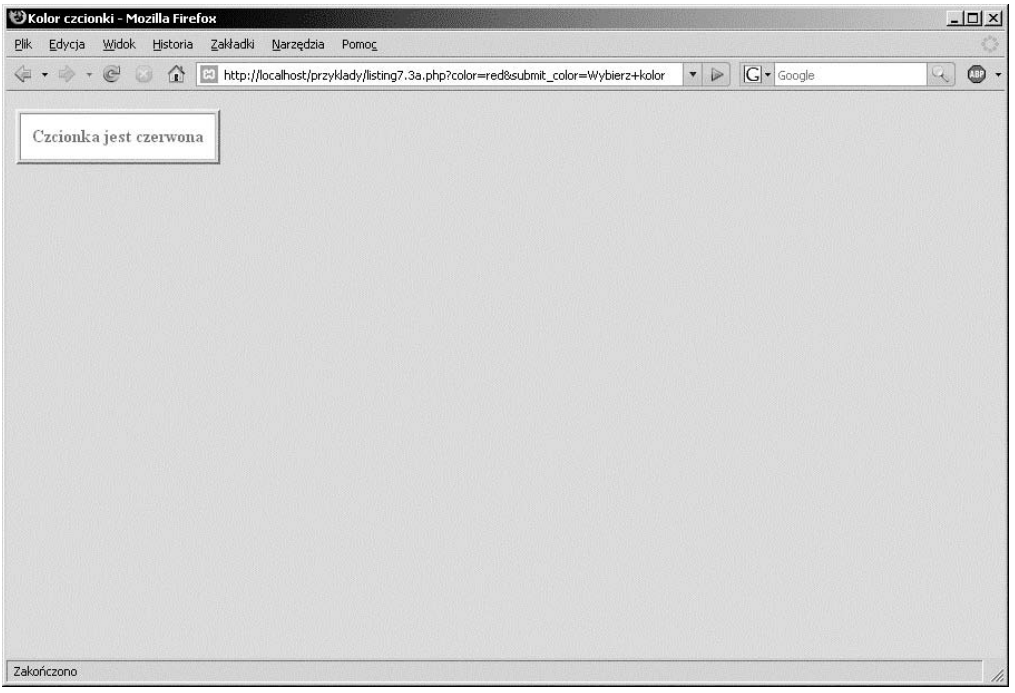
(8) Blok poleceń `default` zostanie wykonany, jeśli nie zostaną dopasowane żadne wcześniejsze etykiety.

Ostatnie polecenie `default` nie jest niezbędne, ale dobrym nawykiem jest jego stosowanie. Polecenie `default` można później zastąpić dodatkową etykietą `case`.

(9) Końcowy nawias klamrowy zamyka blok poleceń konstrukcji `switch`.



Rysunek 7.6. *Formularz HTML przedstawiony na listingu 7.3*



Rysunek 7.7. *Użytkownik zaznaczył przycisk opcji „czerwony". Dane wyjściowe skryptu PHP z listingu 7.3*

7.2. Pętle

Pętle są wykorzystywane do powtarzania wykonywania segmentu kodu aż do chwili spełnienia warunku. Załóżmy, że mamy program odliczający 10, 9, 8,… aż do 1, a następnie wyświetlający komunikat „Start”. W celu odliczenia od 10 do 1 można użyć albo dziesięciu poleceń `print`, albo jednej pętli.

Podstawowe konstrukcje pętli dostarczane przez PHP to:

- `while`
- `for`
- `foreach`
- `do-while`

Pętlę działają nieco inaczej niż konstrukcje `if`. Blok poleceń znajdujący się po instrukcji `if` jest wykonywany jednokrotnie, podczas gdy blok poleceń pętli może być wykonywany wielokrotnie.

7.2.1. Pętla while

Pętla `while` wykonuje blok poleceń tak długo, dopóki wyrażenie po instrukcji `while` zwraca wartość `true`, to znaczy nie-`NULL`, wartość niezerową i nie-`false`. Jeżeli wyrażenie `while` zwraca wartość `true`, wówczas wykonany zostaje blok poleceń znajdujący się po tym wyrażeniu. Po wykonaniu ostatniego polecenia w bloku program powraca z powrotem do wyrażenia `while` i sprawdza, czy wyrażenie nadal zwraca wartość `true`. Jeśli wyrażenie nigdy nie ulega zmianie i zwraca wartość `true`, to pętla będzie wykonywana w nieskończoność. Poprzez zmianę warunku testowego lub opuszczenie pętli `while` za pomocą polecenia `break` programista może określić długość wykonywania pętli. Jeżeli warunek zwraca wartość `false`, wtedy kontrola nad programem przechodzi do pierwszego polecenia znajdującego się za nawiasem klamrowym zamykającym blok poleceń pętli.

W celu kontroli nad wykonywaniem pętli stosowane są polecenia `break` oraz `continue`.

Działanie pętli `while` zostało zaprezentowane na listingu 7.4. Dane wyjściowe listingu pokazano na rysunku 7.8.

FORMAT PĘTLI `WHILE`:

```
while (warunek) {
   polecenia;
   zwiększenie lub zmniejszenie licznika;
}
```

LISTING 7.4.

```
    <html>
    <head>
    <title>Pętla while</title>
    </head>
    <body>
    <h3>Pętla while</h3>
    <font face=arial size='+1'>
(1) <?php
(2)    $i=10; // Inicjalizacja licznika pętli.
(3)    while ($i > 0 ){ // Sprawdzanie warunku pętli.
(4)       echo "$i ";
(5)       $i--; // Zmniejszenie wartości licznika o jednostkę.
(6)    } // Zakończenie bloku pętli.

(7) ?>
    <font size="+3" color="red">
(8) ...Start!
    </font>
    </body></html>
```

OBJAŚNIENIE

(1) W tym wierszu rozpoczyna się program PHP.

(2) Zmienna `$i` zostaje zainicjalizowana z wartością `10`.

(3) Następuje sprawdzenie wyrażenia znajdującego się po słowie kluczowym `while`. Jeżeli wartość zmiennej `$i` jest większa niż `0`, wówczas następuje wykonanie poleceń znajdujących się w bloku. Wartość `false` sprawdzanego wyrażenia (na przykład wartość zmiennej `$i` nie jest większa niż `0`) powoduje opuszczenie bloku i przekazanie kontroli nad programem do wiersza ósmego.

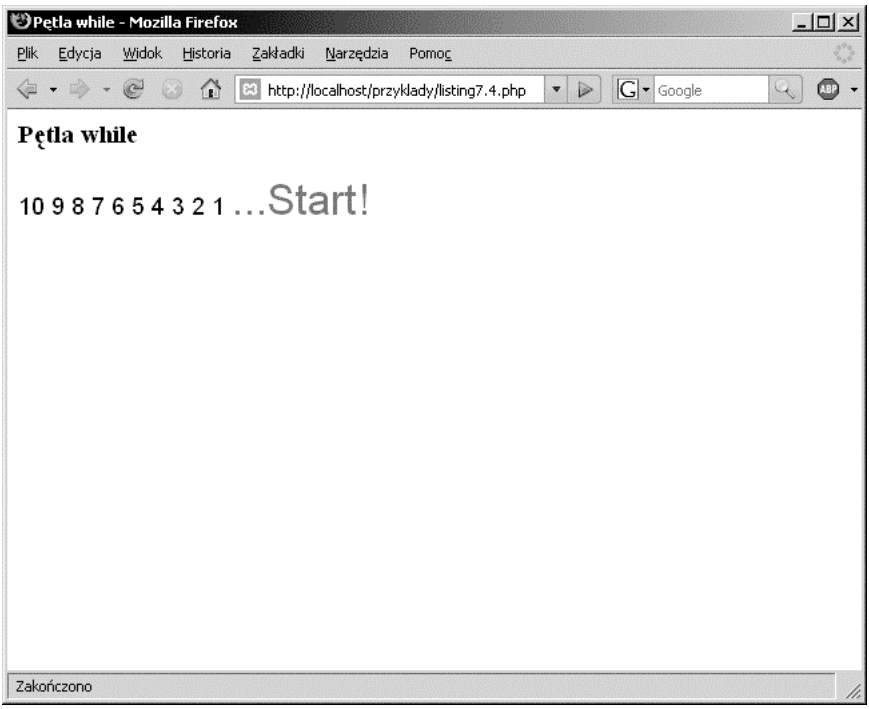
(4) Wartość zmiennej `$i` zostaje wyświetlona w oknie przeglądarki internetowej.

(5) Wartość zmiennej `$i` zostaje zmniejszona o jeden. Ten krok jest bardzo ważny, ponieważ jeśli wartość zmiennej `$i` nigdy nie ulegnie zmianie, to pętla będzie wykonywana w nieskończoność.

(6) Ten nawias klamrowy zamyka blok poleceń pętli `while`. Kontrola nad programem będzie przebiegała tak jak pokazuje strzałka tak długo, aż wyrażenie testowe instrukcji `while` zwróci wartość `true`.

(7) W tym wierszu następuje koniec programu PHP.

(8) Wyświetlenie tego komunikatu HTML następuje już po opuszczeniu pętli.



Rysunek 7.8. *Pętla while. Dane wyjściowe listingu 7.4*

7.2.2. Pętla do-while

Konstrukcja `do-while` wykonuje blok poleceń do czasu, aż warunek nie zwróci wartości `false`. Z powodu swojej struktury przed sprawdzeniem warunku znajdującego się na końcu bloku pętla wykona co najmniej jeden raz polecenia z bloku. Przykład użycia pętli `do-while` został przedstawiony na listingu 7.5. Dane wyjściowe listingu pokazano na rysunku 7.9.

FORMAT PĘTLI `DO-WHILE`:

```
do
   {polecenia;}
while (warunek);
```

LISTING 7.5.

```
<html>
<head>
<title>Pętla do... while</title>
<body bgcolor='f0f8ff'>
<h3>Pętla Do... While</h3>
<font face='arial' size='+1'>
<?php
```

```
(1)    $i=10;
(2)    do {
(3)       echo "$i ";
(4)       $i--;
(5)    }while ( $i > 0 );

    ?>
    </body>
    </html>
```

OBJAŚNIENIE

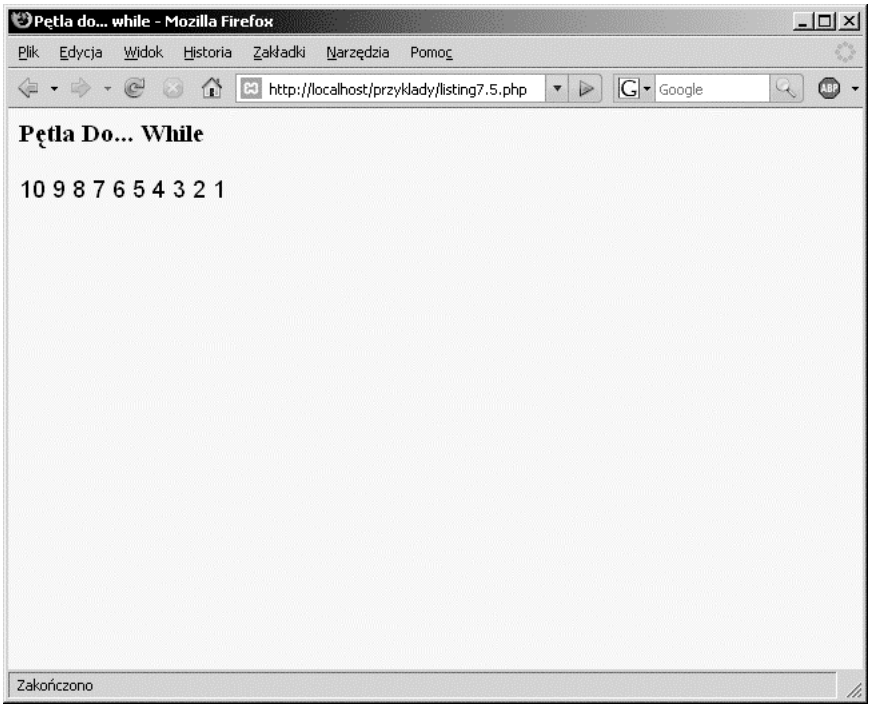
(1) Zmienna `$i` zostaje zainicjalizowana z wartością `10`.

(2) Wejście do bloku poleceń `do`. Blok poleceń zostanie wykonany przed sprawdzeniem wyrażenia `while`. Nawet jeśli się okaże, że wyrażenie `while` zwraca wartość `false`, to blok zostanie wykonany co najmniej raz.

(3) Wartość zmiennej `$i` zostaje wyświetlona w oknie przeglądarki internetowej. Zobacz rysunek 7.9.

(4) Wartość zmiennej `$i` zostaje zmniejszona o jeden.

(5) W tym wierszu następuje sprawdzenie wyrażenia `while`. Jeśli wartością zwrotną jest `true`, wtedy zmienna `$i` jest większa niż `0`. W takim przypadku kontrola nad programem powraca do wiersza drugiego i następuje ponowne wejście do bloku poleceń pętli.



Rysunek 7.9. *Pętla do-while*

7.2.3. Pętla for

Generalnie pętla `for` działa podobnie jak pętla `while`, jest tylko bardziej skondensowana. Pętla `for` składa się ze słowa kluczowego `for`, po którym znajdują się trzy wyrażenia rozdzielone średnikami i ujęte w nawiasy. Dowolne z tych wyrażeń lub nawet wszystkie mogą zostać pominięte, ale dwa średniki muszą zawsze pozostać. Pierwsze wyrażenie jest używane do ustalenia początkowej wartości zmiennej i jest wykonywane jednorazowo. Drugie wyrażenie jest używane do sprawdzenia, czy wykonywanie pętli należy kontynuować czy przerwać. Trzecie wyrażenie natomiast służy do uaktualniania zmiennych pętli, na przykład zmniejszenia lub zwiększenie licznika o jednostkę, co zwykle wpływa na czas wykonywania pętli.

Wykorzystanie pętli `for` zostało przedstawione na listingu 7.6. Dane wyjściowe listingu pokazano na rysunku 7.10.

FORMAT PĘTLI `FOR`:

```
for (wyrażenie1;wyrażenie2;wyrażenie3)
   {polecenie(a);}
for (inicjalizacja; warunek; zmniejszenie/zwiększenie)
   {polecenie(a);}
```

Przedstawiony format odpowiada poniższej pętli `while`:

```
wyrażenie1;
while (wyrażenie2)
   {blok; wyrażenie3};
```

LISTING 7.6.

```
<html>
<head>
<title>Pętla for</title>
</head>
<body bgcolor="lightblue">
<h3>Pętla for</h3>
<font face='arial' size='+1'>
<?php
(1)   for( $i = 0; $i < 10; $i++ ){
(2)      echo "$i ";
(3)   }

?>
</body>
</html>
```

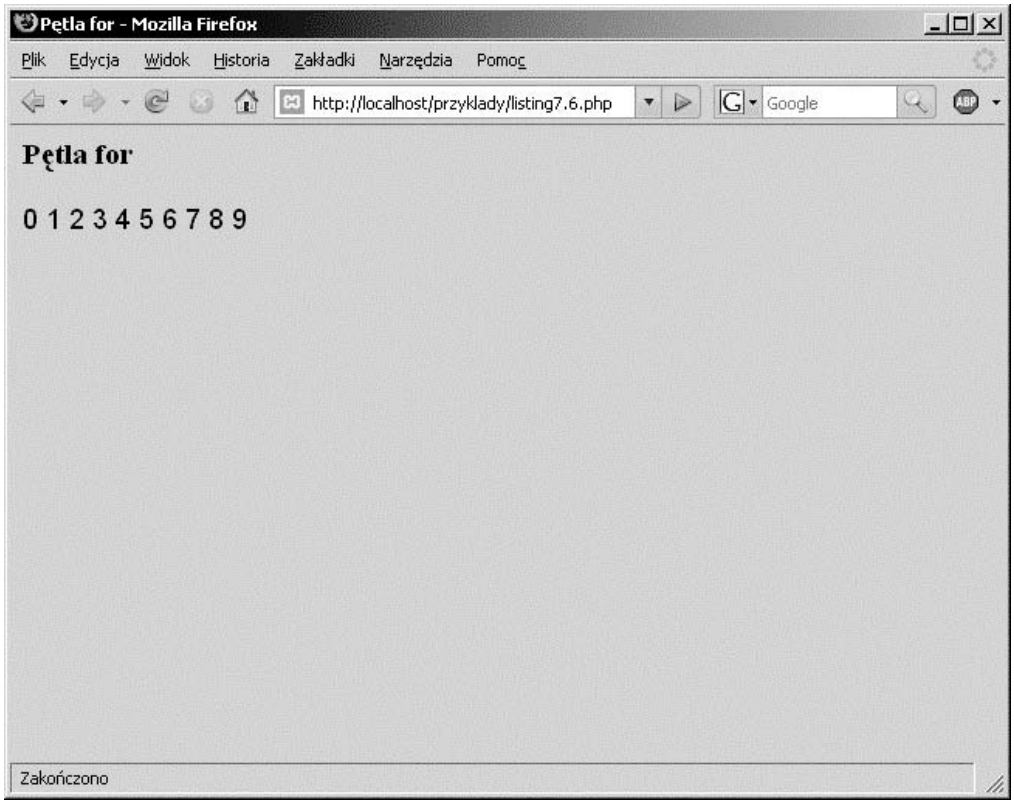
OBJAŚNIENIE

(1) W tym wierszu następuje wejście do pętli `for`. Wyrażenie rozpoczyna się od kroku ❶, inicjalizacji zmiennej `$i` z wartością zero. Ten krok jest wykonywany jednorazowo. Drugie wyrażenie, krok ❷, sprawdza, czy wartość zmiennej `$i`

jest mniejsza niż `10` i jeśli tak, to następuje wykonanie poleceń w bloku — krok ❸. Kiedy wszystkie polecenia w bloku zostaną wykonane, kontrola powraca ponownie do pętli `for`, do trzeciego (ostatniego) wyrażenia (krok ❹). Wartość zmiennej `$i` zostaje zwiększona o jednostkę i następuje ponowne sprawdzenie wyrażenia z kroku ❷. To działanie jest teraz krokiem ❺. Jeżeli wynikiem sprawdzenia będzie wartość `false`, to działanie pętli będzie zakończone, natomiast wartość `true` powoduje wejście do bloku poleceń i ich wykonanie.

(2) Wartość zmiennej `$i` zostaje wyświetlona w oknie przeglądarki internetowej.

(3) Zamykający nawias klamrowy oznacza koniec pętli `for`.



Rysunek 7.10. *Pętla for*

Pętla `for` i powtarzające się pola formularza. Formularz HTML przedstawiony na listingu 7.7 zawiera zbiór pięciu pól wyboru z powtarzającymi się nazwami. Jedynym fragmentem nazwy, który różni się od innych, jest cyfra znajdująca się na końcu nazwy. Dzięki wykorzystaniu pętli `for` oraz zmiennej zmiennych[1] istnieje możliwość dynamicznego utworzenia nazw pól oraz dostęp do ich wartości.

[1] Zmienne zmiennych zostały szczegółowo przedstawione w rozdziale 4.

LISTING 7.7.

```
(Formularz HTML)
    <html><head><title>Wielokrotne wybory</title></head>
    <body bgcolor="aqua">
    <form action="listing7.7a.php" method="post">
    <b>Wybierz miejsce spędzenia wakacji:</b>
       <br />
       <input type="checkbox" name="place1" value="New York">Nowy Jork
       <br />
       <input type="checkbox" name="place2" value="Chicago">Chicago
       <br />
       <input type="checkbox" name="place3" value="London">Londyn
       <br />
       <input type="checkbox" name="place4" value="Tokyo">Tokio
       <br />
       <input type="checkbox" name="place5" value="San Francisco"
          Checked>San Francisco
       <p>
       <input type="submit" value="Wybierz">
       <input type="reset" value="Wyczyść">
       </p>
    </form>
    </body>
    </html>
--------------------------------------------------------------------------------
(Skrypt PHP)
    <html><head><title>Pętla for oraz zmienne zmiennych</title></head>
    <body bgcolor="000099">
    <font face="arial" size="+1">
    <table border="1" bordercolor="white" cellpadding="3">
    <tr>
       <td bgcolor="00ff66" align="center">Elementy listy</td>
       <td bgcolor="00ff66" align="center">Zaznaczone wartości</td>
    </tr>

    <?php
    extract($_REQUEST);
         // Używanie tablicy REQUEST wymaga zachowania ostrożności
(1)    for( $i=1; $i <= 5; $i++){
(2)       $temp = "place$i";
(3)       echo "<tr><td bgcolor='00ff99'>$temp</td>";
(4)       echo "<td bgcolor='00ffcc'>${$temp}</td></tr>";
       }

    ?>
    </table>
    </font>
    </body>
    </html>
```

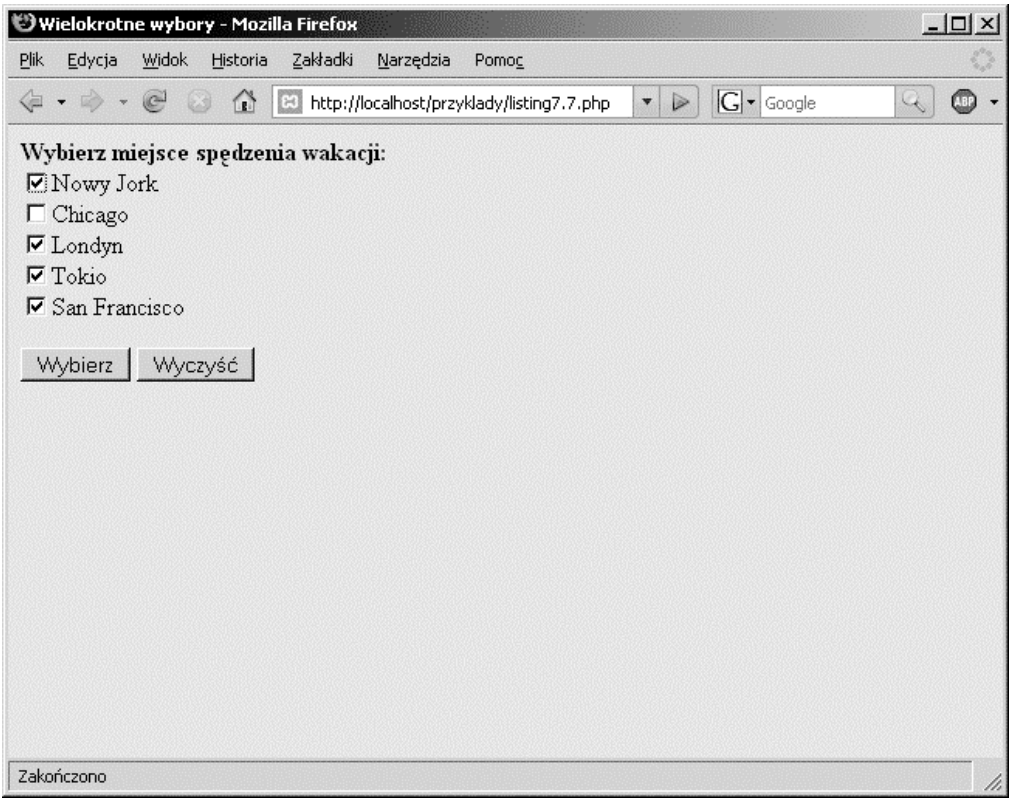
OBJAŚNIENIE

(1) Po wyodrębnieniu danych wejściowych użytkownika wysłanych z formularza pokazanego na rysunku 7.11 następuje wykonanie pętli `for` dla każdego z pięciu elementów pól wyboru, które mogły zostać wybrane.

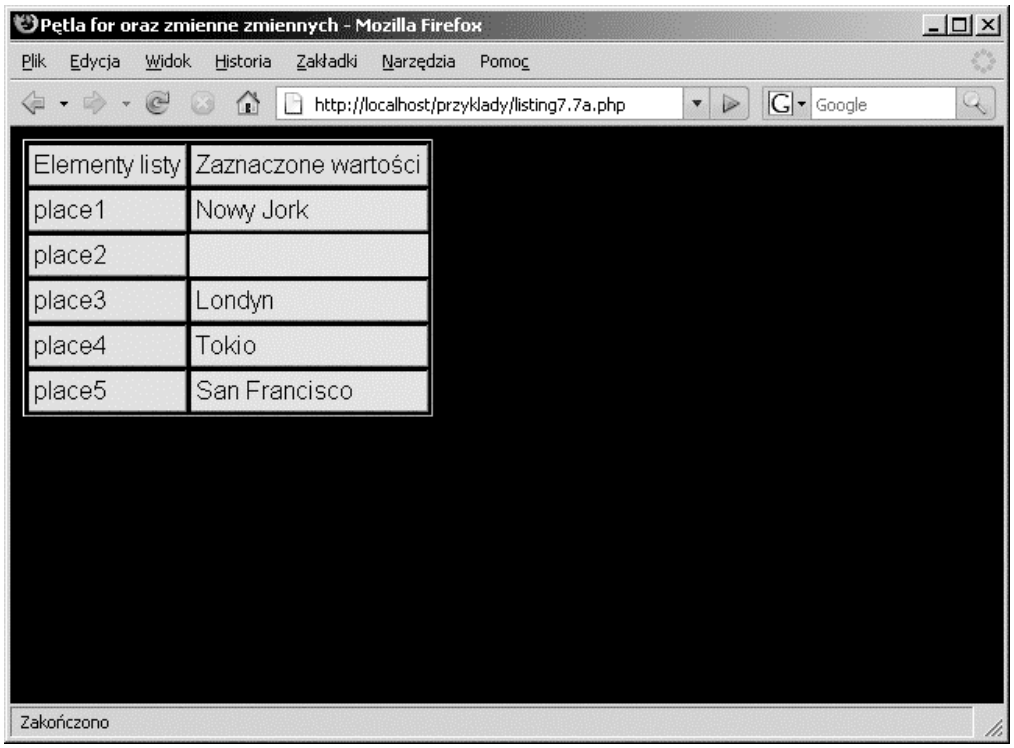
(2) W trakcie pierwszego wykonania pętli `for` zmiennej `$temp` zostaje przypisany ciąg tekstowy `"place$i"`, ponieważ wartość początkowa zmiennej `$i` wynosi `1`.

(3) Wartość zmiennej `$temp` zostaje umieszczona w komórce tabeli.

(4) Zmienna `$temp` (na przykład `"place$i"`) jest teraz traktowana jako „zmienna zmiennej", w której `place1` staje się `$place 1`, czyli wartość zaznaczona dla tego pola wyboru. Jeżeli użytkownik nie zaznaczył tego pola wyboru, wartością będzie `null`. Wartość zaznaczonego pola wyboru będzie wstawiona do tabeli po prawej stronie nazwy danego pola. Pętla `for` zostanie wykonana pięciokrotnie, zwiększając wartość zmiennej `$i` o jednostkę podczas każdego wykonania pętli. W wyniku tego działania otrzymamy dane wyjściowe pokazane na rysunku 7.12.



Rysunek 7.11. *Formularz HTML przedstawiony na listingu 7.7 wraz z zaznaczonymi polami wyboru*



Rysunek 7.12. *Dane wyjściowe programu PHP z listingu 7.7*

7.2.4. Pętla foreach

Pętla `foreach` została zaprojektowana do pracy z tablicami i działa jedynie z nimi. Szczegółowe omówienie tablic zostanie przedstawione w rozdziale 8., ale ponieważ w niniejszym rozdziale omawiane są pętle, tak więc również wzmianka o pętli `foreach` powinna się tu znaleźć.

Tablica jest zbiorem elementów, na przykład tablica liczb lub ciągów tekstowych. Wyrażenie pętli składa się z nazwy tablicy, słowa kluczowego `as` oraz zmiennej zdefiniowanej przez użytkownika, która będzie przechowywała kolejne wartości tablicy podczas wykonywania pętli. Pętla `foreach`, jak wskazuje jej nazwa[2], działa kolejno na każdym elemencie tablicy, przesuwając się od lewej do prawej strony, aż do przetworzenia wszystkich elementów tablicy. Za wyrażeniem pętli znajduje się blok poleceń, który zostanie wykonany dla każdego elementu w wyrażeniu.

[2] Słowo *foreach* oznacza dosłownie „dla każdego” — *przyp. tłum.*

FORMAT PĘTLI FOREACH**:**

```
$nazwa_tablicy = array(element1, element2, element3, ...);
foreach ($nazwa_tablicy as $wartość) {
   wykonanie poleceń na wartości elementu;
}
```

Przykład:

```
$elementy = array('diament', 'łopata', 'kij', 'serce'); // Tablica.
foreach ($elementy as $rodzaj) {
   echo $rodzaj . '<br />'; // Wyświetla: diament
                                          łopata
                                          kij
                                          serce
}
```

7.2.5. Sterowanie działaniem pętli za pomocą poleceń break oraz continue

Polecenia sterujące pętlą `break` oraz `continue` są używane albo do opuszczenia pętli, albo do wczesnego sprawdzania warunku, to znaczy przed osiągnięciem zamykającego nawiasu klamrowego bloku znajdującego się za konstrukcją pętli (zobacz tabela 7.1).

Tabela 7.1. *Polecenia break oraz continue*

Polecenie	Działanie polecenia
`break`	Powoduje opuszczenie pętli `for`, `foreach`, `while` oraz `do-while` i przejście do pierwszego polecenia znajdującego się za blokiem pętli. Polecenie `break` przyjmuje opcjonalny argument numeryczny informujący o liczbie zagnieżdżonych struktur, które polecenie może opuścić.
`continue`	Przekazuje kontrolę pętli bezpośrednio na początek pętli i powoduje ponowne sprawdzenie warunku pętli. Jeśli wartością zwrotną warunku pętli jest `true`, wtedy następuje wejście do bloku pętli. Polecenie `continue` przyjmuje opcjonalny argument numeryczny, informujący o liczbie poziomów pętli, które mogą być „przeskoczone”.

Na listingu 7.8 zostało przedstawione użycie polecenia `break` do opuszczenia pętli na podstawie tego samego warunku. Dane wyjściowe listingu pokazano na rysunku 7.13.

LISTING 7.8.

```
<html><head><title>Opuszczenie pętli</title></head>
<body bgcolor="indigo">
<p>
<table border="1" bordercolor="white" cellpadding="3" bgcolor='thistle'>
<caption><font color="white">Zamrożenie!</font></caption>
<tr><th>Celsjusz</th><th>Farenheit</th><tr>
```

```
    <?php
(1)    $C=-10;
(2)    while($C < 100){
(3)       $F = ( $C * 1.8) + 32;
(4)       print "<tr><td><b>$C</td><td><b>$F</td>";
(5)       if ( $F == 32 ){ // Opuszczenie pętli.
(6)          break;
          }
(7)       $C+=2;
(8)    }
    ?>
    </tr>
    </table>
    </body>
    </html>
```

OBJAŚNIENIE

(1) Zmienna `$C` zostaje zainicjalizowana wraz z wartością `-10`.

(2) Sprawdzenie wyrażenia pętli `while`. Czy wartość zmiennej `$C` jest mniejsza niż `100`? Jeżeli tak, następuje wejście do bloku pętli w wierszu trzecim.

(3) Ta formuła powoduje konwersję stopni Celsjusza na stopnie Fahrenheita.

(4) Wartość zmiennej `$F` zostaje wyświetlona w komórce tabeli HTML.

(5) Jeżeli wartość temperatury w stopniach Fahrenheita jest równa `32`, wówczas kontrola nad programem przechodzi do wiersza szóstego.

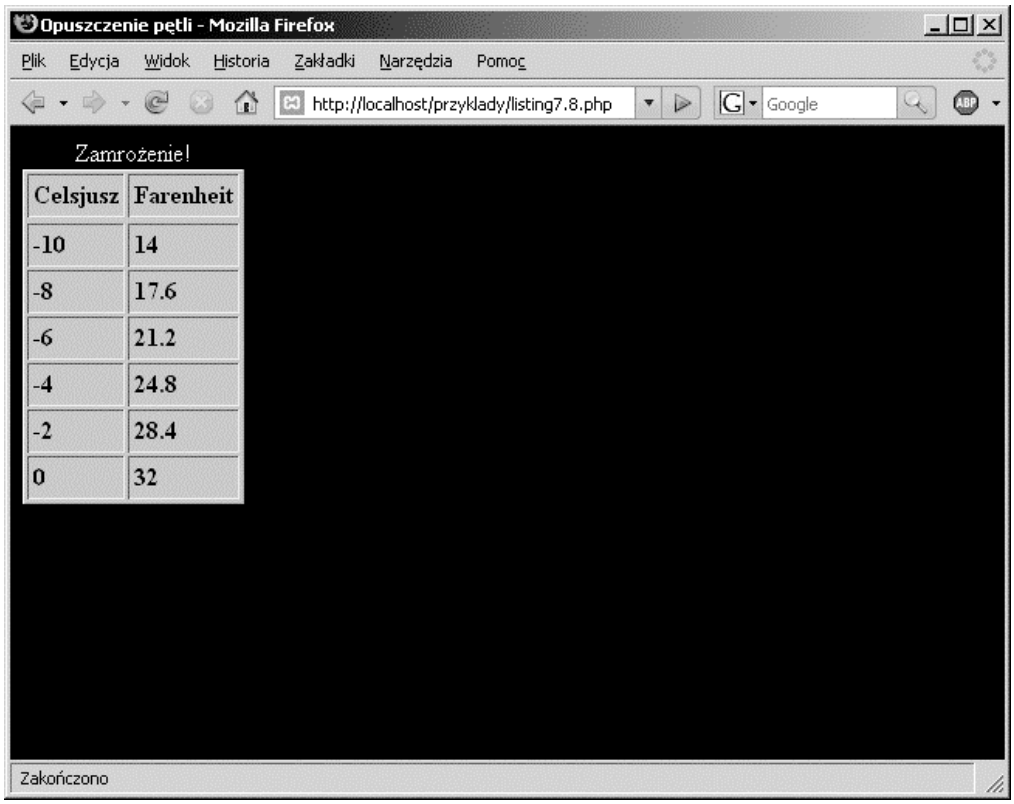
(6) Polecenie `break` powoduje zatrzymanie wykonywania poleceń w bloku i przejście do polecenia znajdującego się za wierszem dziewiątym.

(7) Wartość zmiennej `$C` zostaje zwiększona o dwa podczas każdego wykonania pętli.

(8) Ten wiersz oznacza koniec bloku pętli. Dopóki warunek pętli w wierszu pierwszym zwraca wartość `false` lub nie nastąpi wykonanie polecenia `break`, program będzie kontynuował wykonywanie pętli.

Pętle zagnieżdżone. Pętla znajdująca się wewnątrz innej pętli jest pętlą zagnieżdżoną. Najczęstszym zastosowaniem pętli zagnieżdżonych jest wyświetlanie danych w wierszach i kolumnach. W takiej sytuacji jedna pętla obsługuje wiersze, natomiast druga jest odpowiedzialna za obsługę kolumn. Po zainicjalizowaniu i sprawdzeniu warunku pętli zewnętrznej następuje pełne wykonanie wszystkich cykli pętli wewnętrznej, po czym pętla zewnętrzna rozpoczyna się ponownie od miejsca, w którym się zatrzymała. Pętla wewnętrzna posuwa się szybciej od zewnętrznej. Pętle mogą być zagnieżdżone na dowolnej liczbie poziomów, ale istnieją sytuacje, w których należy przerwać działanie pętli ze względu na pewien warunek.

Działanie pętli zagnieżdżonych zostało przedstawione na listingu 7.9. Dane wyjściowe listingu pokazano na rysunku 7.14.



Rysunek 7.13. *Sterowanie wykonywaniem pętli. Dane wyjściowe listingu 7.8*

LISTING 7.9.

```
    <html><head><title>Pętle zagnieżdżone</title></head>
    <body bgcolor="lightgreen">
    <font face="arial" size="+1">
    <div align="center">
    <b>

    <?php
(1)    $character = '**';
(2)    echo '*';
(3)       for ($row=0; $row < 10; $row++){
(4)          for ($col=0; $col < $row; $col++){
                echo $character;
             }
(5)       echo "\n<br />";
          }
       echo "| |\n<br />";
    ?>
    <font color='red'>Wesołych świąt Bożego Narodzenia!</font></font><br />
    </div>
    </body>
    </html>
```

OBJAŚNIENIE

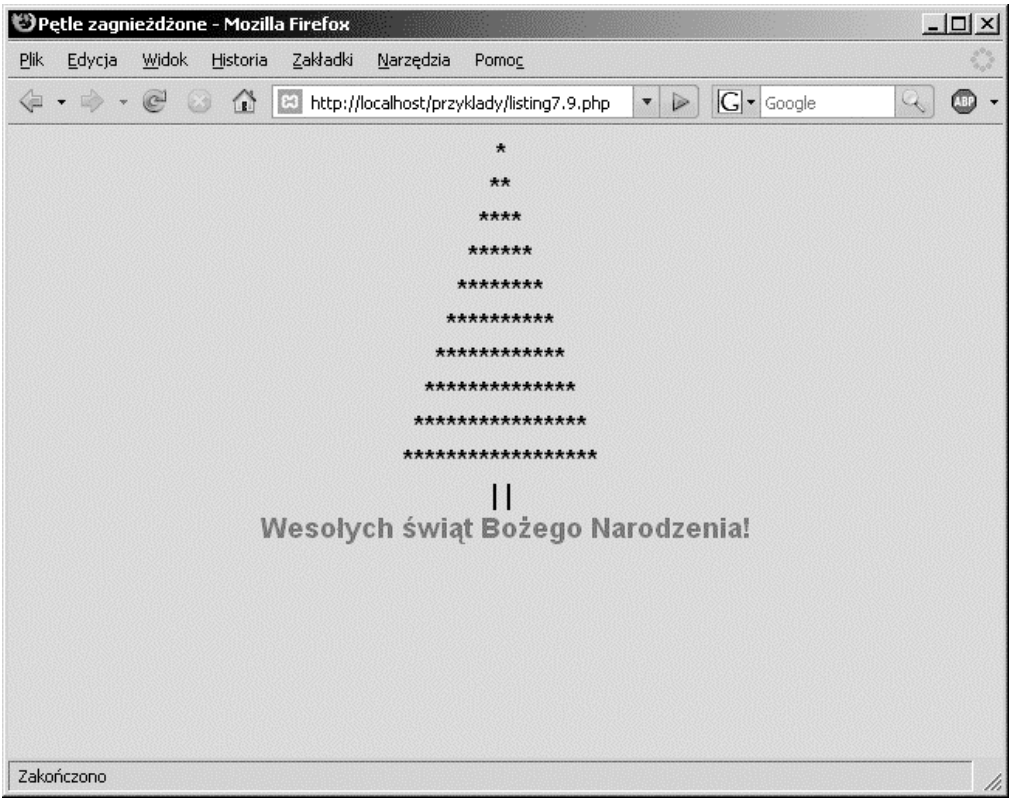
(1) Zmiennej `$character` zostaje przypisany ciąg tekstowy `'**'`.

(2) Wyświetlenie pojedynczego znaku *.

(3) Wejście do zewnętrznej pętli `for`. Zmienna `$row` zostaje zainicjalizowana z wartością `0`. Jeżeli wartość zmiennej `$row` jest mniejsza niż `10`, wówczas następuje wejście do bloku pętli (w nawiasach klamrowych), to znaczy przejście do czwartego wiersza.

(4) Wejście do wewnętrznej pętli `for`. Zmienna `$col` zostaje zainicjalizowana z wartością `0`. Jeżeli wartość zmiennej `$row` jest mniejsza od wartości zmiennej `$row`, wówczas następuje wejście do bloku pętli i wyświetlenie znaku *. Następnie wartość zmiennej `$col` zostaje zwiększona o jednostkę, ponownie sprawdzenie i jeśli nadal jest mniejsza od wartości zmiennej `$row` — następuje wejście do bloku pętli i wyświetlenie kolejnych znaków **. Kiedy pętla zostanie zakończona, na ekranie będzie wyświetlony rząd znaków * i nastąpi ponowne wykonanie poleceń zewnętrznej pętli.

(5) Po zakończeniu działania pętli zewnętrznej nastąpi wykonanie wiersza powodującego wyświetlenie znaku nowego wiersza.



Rysunek 7.14. *Pętle zagnieżdżone: wiersze i kolumny. Dane wyjściowe listingu 7.9*

Jeżeli wykorzystywane są polecenia nadzorujące działanie pętli, takie jak `break` oraz `continue`, wówczas zazwyczaj kontrola jest kierowana do pętli najbardziej zewnętrznej. Istnieją sytuacje, w których może być niezbędne przekazanie kontroli do dowolnej pętli wewnętrznej. W takich przypadkach dostępny jest drugi opcjonalny argument poleceń `break` oraz `continue` pozwalający na wybór pętli opuszczanej lub tej, od której będzie kontynuowane działanie. Wspomniany argument przedstawia pętlę, do której zostanie przekazana kontrola. Przykładowo polecenie `break 2` opuści bieżącą pętlę, pętlę 1 oraz kolejną pętlę — pętlę 2. Przykładowy skrypt został przedstawiony na listingu 7.10.

LISTING 7.10.

```
(Przykładowy skrypt)
    <?php
(1) while(1) {
(2)    < Kontynuowanie wykonywania programu >
(3)    while(1) {
(4)       if (<wyrażenie wynosi true>) { break 2;}
          <Kontynuowanie wykonywania programu >
(5)       while(1) {
(6)          if (<wyrażenie wynosi true>) { continue 3;}
             < Kontynuowanie wykonywania programu >
          }
       }
    }
(7) print 'Opuszczono wszystkie pętle.<br />';
    ?>
```

7.3. Podsumowanie rozdziału

7.3.1. Co należy wiedzieć?

Po zapoznaniu się z materiałem przedstawionym w niniejszym rozdziale Czytelnik powinien być w stanie odpowiedzieć na poniższe pytania:

1. Co to jest blok? W jaki sposób polecenia są zagnieżdżane w blokach?
2. W jaki sposób funkcjonują struktury sterujące takie jak `if`, `else` oraz `elseif`?
3. Jak można używać polecenia `switch`?
4. W jaki sposób można używać pętli `while`, `do-while`, `for` oraz `foreach`?
5. Co powoduje, że pętla jest wykonywana w nieskończoność?
6. Jak można nadzorować pętle za pomocą poleceń `break` oraz `continue`?
7. W jaki sposób można zagnieżdżać pętle?

7.3.2. Co dalej?

W kolejnym rozdziale zostaną przedstawione zagadnienia dotyczące tablic liczbowych i asocjacyjnych oraz wielu użytecznych wbudowanych funkcji PHP przeznaczonych do operowania na tablicach. Poznamy również wbudowane superglobalne tablice PHP oraz sposoby ich używania.

ĆWICZENIA

1. Instrukcje warunkowe `if-else`.

a) Napisz formularz HTML, który będzie prosił użytkownika o podanie swojego wieku. Jeśli podany wiek będzie niższy niż 21 lat, wówczas użytkownika należy poinformować, że jest zbyt młody na spożywanie alkoholu. Jeżeli podany wiek jest wyższy niż 21 lat, wtedy należy poinformować użytkownika, że jest za stary na spożywanie alkoholu i prowadzenie samochodu.

b) Napisz formularz HTML proszący użytkownika o podanie wyników z trzech testów. Dostępny zakres wyników to od 0 do 100.

Napisz program PHP sprawdzający, czy podane przez użytkownika wyniki są wartościami liczbowymi (zobacz: funkcja `is_numeric()`). Oblicz średnią ocenę z podanych trzech testów, używając poniższej skali. Do określenia oceny użyj konstrukcji `if-elseif`.

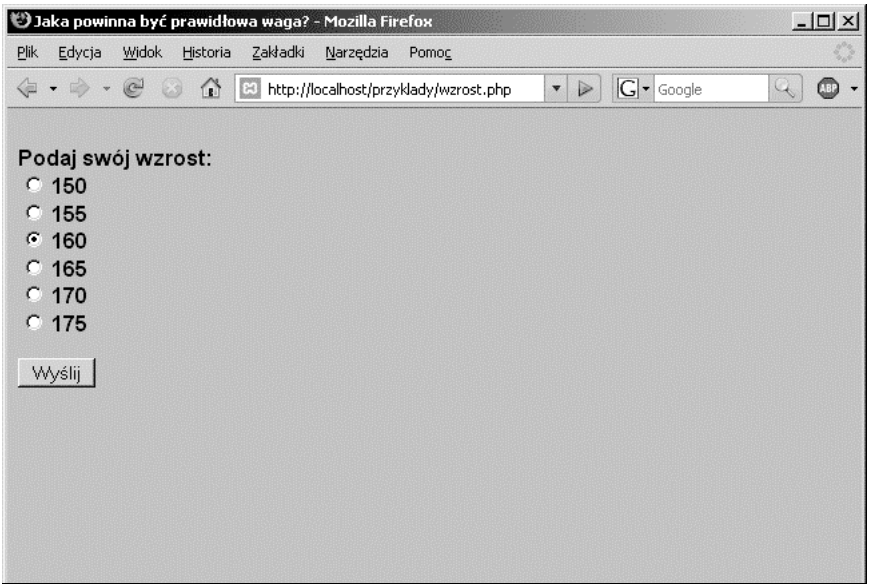
```
90 - 100  = 5
80 - 89 = 4
70 - 79 = 3
60 - 69 = 2
0 - 59 = 1
```

W przeglądarce internetowej należy wyświetlić średnią liczbę punktów z trzech testów oraz przyznaną ocenę. Jeżeli użytkownik otrzymuje ocenę 5, wówczas dane wyjściowe powinny być wyświetlone za pomocą nagłówka pierwszego stopnia HTML, czyli `<h1>`. W przypadku oceny 4 należy zmniejszyć stopień nagłówka o jeden i tak aż do oceny 1. Jeżeli użytkownik otrzymał ocenę 1, wtedy należy również wyświetlić komunikat „Powinieneś się bardziej skoncentrować” za pomocą czerwonego znacznika `<h5>`.

c) Utwórz formularz proszący użytkownika o podanie miasta docelowego podróży z listy przycisków opcji. Dostępnymi miastami powinny być: San Francisco, Nowy Jork, Londyn, Paryż, Honolulu oraz Tokio.

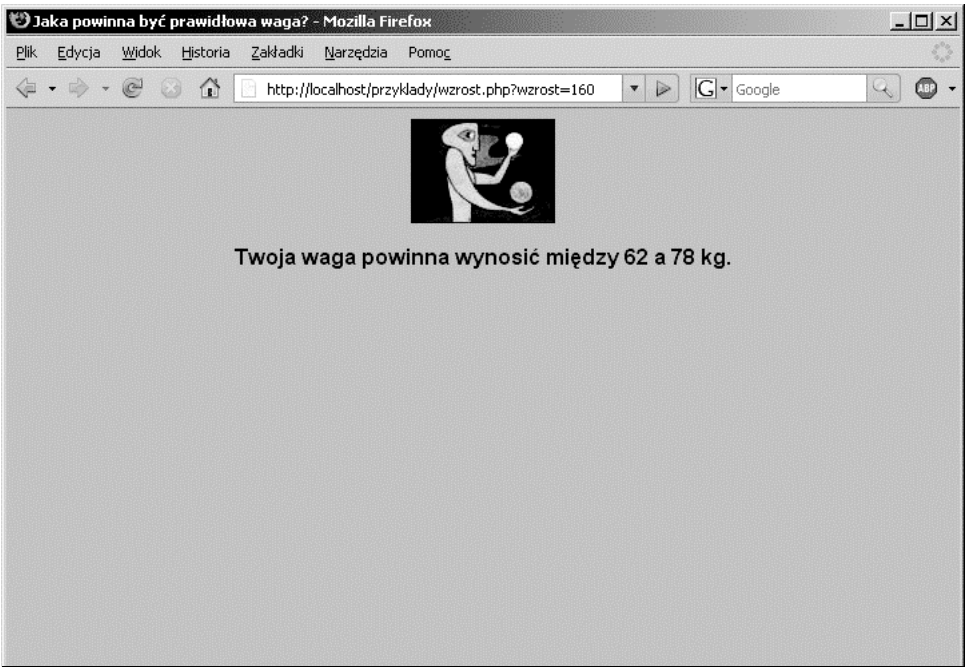
Napisz skrypt PHP używający polecenia `switch` do przejścia przez każde miasto. Na podstawie wyboru użytkownika skrypt powinien wyświetlać komunikat, na przykład „Witamy w San Francisco. Nie zapomnij o swoim sercu i marynarce!” lub „Witamy, pójdźmy na spacer w lewą stronę!”.

d) Utwórz formularz PHP, który będzie używał przycisków opcji, tak jak pokazano na rysunku 7.15.



Rysunek 7.15. *Jaka powinna być prawidłowa waga?*

Utwórz skrypt PHP wykorzystujący polecenia `case` w celu wyświetlenia danych wyjściowych z uwzględnieniem wybranego przycisku opcji. Wybierz dowolną grafikę. Dane wyjściowe PHP powinny mieć następującą postać:



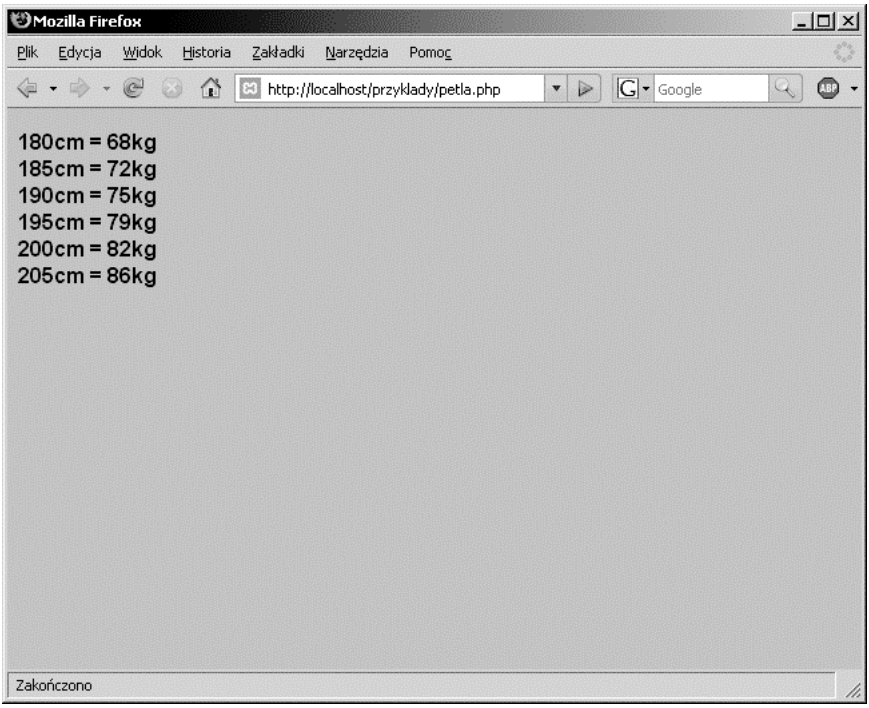
Rysunek 7.16.

W celu utworzenia warunków na podstawie zaznaczonego przez użytkownika przycisku należy użyć następującego schematu:

Wysokość	Prawidłowa waga
150	48 – 64
155	55 – 71
160	62 – 78
165	69 – 85
170	76 – 92
175	83 – 99

2. Używanie pętli.

a) Korzystając z przedstawionego w poprzednim ćwiczeniu schematu odczytujemy, że każde 5 cm wzrostu powoduje wzrost wagi o 3,5 kg. Napisz pętlę, która będzie wyświetlała wzrost oraz wagę począwszy od 180 cm do 205 cm w przedziałach co 5 cm (na przykład 180 cm, 185 cm, 190 cm itd.). Podczas każdego przejścia pętli w przeglądarce internetowej należy wyświetlić wynik, jak pokazano na rysunku 7.17.



Rysunek 7.17. *Wzrost oraz waga*

b) Dysponując początkowymi wagami, utwórz tabelę HTML zawierającą wzrost, wagę początkową oraz wagę krańcową. Załóżmy, że waga krańcowa dla 180 cm wynosi 92 kg i wrasta o 2 kg dla każdych 5 cm wzrostu (na przykład dla 185 cm waga krańcowa to 94 kg itd.).